MASTER OF SCIENCE DEGREE/DEGREE WITH HONOURS IN
Embedded Intelligent Systems
Final Year Project Report
Department of Electronic, Communication and Electrical Engineering
University of Hertfordshire

INVERTED PENDULUM CONTROL

Report by
SHAREEF MOHD ASLAM
Supervisor
DR. DAVID LEE
Date
SEPTEMBER-2007

University of Hertfordshire

DECLARATION STATEMENT

I certify that the work submitted is my own and that any material derived or quoted from the published or unpublished work of other persons has been duly acknowledged (ref. UPR AS/C/6.1, Appendix I, Section 2 – Section on cheating and plagiarism)

Student Full Name: Mohd Aslam Shareef

Student Registration Number: 06144184

Signed: ……

Date: 16 January 2008

University of Hertfordshire

ABSTRACT

Inverted pendulum is an application of a Mechatronics. Mechatronics means combination of different department mechanical engineering, electronic engineering, computer science, control system and electrical engineering. The two challenging inverted pendulum are arm-driven and rotational system, whose aim is to balance a pendulum on upright position using feedback control.

This report mainly describes how to close the inverted pendulum hardware loop using encoder. Detailed working principles of each part or component used in this project are discussed. Solutions for technical problems are discussed in detail. How inverted pendulum hardware can be closed without using encoders by choosing some other alternate options like timers, used to generate a pulses which is quite similar to the Incremental encoder module output waveforms for testing purposes.

A detailed description of how to configure and program a Z8 encore! MCU Development board for a specific task like timers, interrupts and GPIO port initialization are discussed. The objective has been achieved in hardware, the Z8 controller acts as a interface between host PC and inverted pendulum hardware to balance a pendulum in dynamic state.

Also, this report focuses on the FIS logic controller, fuzzy sets, membership functions and fuzzy rules.

In the end, discussion on project time/budget management and project conclusions are made. Some of the potential further works are highlighted in the report.

University of Hertfordshire

ACKNOWLEDGEMENTS

All Praises to God Almighty for the benefactions of his mercy to me all the time I wish to acknowledge the generosity and co-operation of my project supervisor, ***Dr. DAVID LEE*** for his continued assistance and guidance towards successful completion of the project. His superb guidance has leaded me on the way of learning.

I would like to express my gratitude to Mr. John Wilmot from the university workshop (Lab C460); He had been very friendly and helped me with all my components ordering and he assisted me in building the project hardware. This project couldn't be completed without him.

Also I would like to express my gratitude to my parents, and all others who have shown a great deal of affection toward me through out my ordeals. Without their support I wouldn't imagine myself in this position.

I would also like to thanks my brother and sisters for there encouragements and support.

Lastly a special thanks to all my friends for their support and enthusiasm and also i like to thank the following department and institutions:

1. Manufacturing Department, for there co-operation, University of Hertfordshire.
2. The department of Electrical Electronics and Communication Engineering, University of Hertfordshire, for their co-operation.

University of Hertfordshire

TABLE OF CONTENTS

ABSTRACT i
ACKNOWLEDGEMENTS i
GLOSSARY iv

CHAPTER 1 : INTRODUCTION 1
1.1 Project Background 1
1.1.1 Inverted Pendulum System 1
1.1.2 Inverted Pendulum System Modelling methology 2
1.2 Project Aims and Objectives 2
1.3 Overview of the Project 3
1.4 Organisation of Report 4

CHAPTER 2 : INVERTED PENDULUM HARDWARE 5
2.1 Pole, Shaft and First Link 5
2.2 Two-Phase bipolar stepper motar 6
2.3 L298 Full-Bridge driver from National Semiconductor 7
2.4 HEDS-9140 Incremental Encoder From Hewlett-Packard 11
2.4.1 HEDS-9140 Incremental Encoder Characterstics 11
2.4.2 HEDS-5140 Code Wheel 15
CHAPTER 3 : Programming and configuration of the Z8 controller 17
3.1 Overview and Configuration of the Zilog Z* Encore! MCU Conntroller 17
3.2 Configuration of GPIO 18
3.3 Configuration of Interrupt 23
3.3.1 Interrupt port for Inilialization 24
3.3.2 Set Interrupt Priority 25
3.3.3 Interrupt Edge Selection Register 26
3.3.4 Interrupt Port Select 27
3.4 Configuration of Timer 27
CHAPTER4 : Fuzzy Interference System 31
4.1 FIS Introduction 31
4.2 Design of Fuzzy Controller 32

University of Hertfordshire

4.2.1 Fuzzification ……. 32
4.2.2 Rule Evaluatioin ……... 33
4.2.3 Defuzzification ……..34

CHAPTER 5: Implementation and Testing ……36
5.1 Incremental Encoder Module ……... 36
5.2 Close Inverted Pendulum Hardware ……. 36
CHAPTER 6 : Project Management ……39
6.1 Time Management……..39
6.1.1 Extra time spend on selecting components and buillding inverted pendulum hardware……..39
6.1.2 Extra time spend on programming, configuring the Z8 encore Microcontroler development board ……. 39
6.2 Budget Management ……. 40
6.3 Equipments and resources used in laboratory …….. 41
CHAPTER 7: Conclusion and Further work ……. 42
7.1 Conclusion ……... 42
7.2 Acheivements……42
7.3 Overall Comment on this project……..43
7.4 Potenial Further work……44
7.4.1 Problems and difficulties……44
7.4.2 Further work……...44
REFERENCES
Bibliography
Appendix

University of Hertfordshire

GLOSSARY

FIS	Fuzzy Interference System
GPIO	General Purpose Input Output
IC	Integrated Circuit
MCU	Micro-controller-unit
MF	Membership Function
PWM	Pulse Width Modulation
PID	Proportional Integral derivative
LED	Light Emitting Diode
PAOUT	Port A Output
PDIN	Port D Input

University of Hertfordshire

1. INTRODUCTION

In this chapter, discussions will be on project background, project overview, project aims and objectives

1.1 Project Background

1.1.1 Inverted Pendulum system

Inverted pendulum system is widely used in automatic control systems. Inverted Pendulum is a non-linear, unstable and multi-variable system.

This project is mainly about rotational inverted pendulum hardware, design, control and fuzzy interference system (FIS). An inverted pendulum system typically consists of two links rotating about an axis. First link, driven by a stepper motor rotates in horizontal axis to balance a second link (pendulum or pole) which rotates freely in vertical axis. The inverted pendulum system typically consists of four inputs pole angle, pole position, angular velocity and speed and single output torque. By monitoring the current input values the controller generates an appropriate direction (Clockwise or Anti-clockwise) of torque to the first link (Horizontal link) so that the second link (pole) can be balanced in the upright position as shown in figure 1.1.1 below.

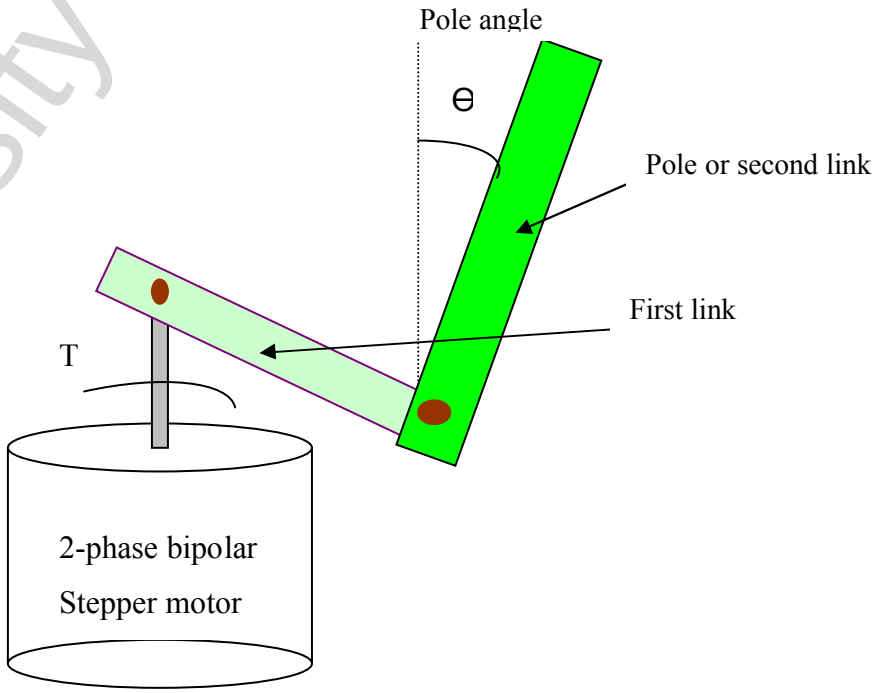


Figure 1.1.1 Inverted Pendulum

University of Hertfordshire

1.1.2 Inverted pendulum system modelling methodology:

Inverted pendulum system is one of the most common controlling methodologies in robotics, industries and in control areas. The inverted pendulum is famous because many variations of the system represent different kind of robotics arms and balancing. A lot of research on control of inverted pendulum system using different methodologies has been done. After reading technical papers, I come to know that there are so many controllers to balance an inverted pendulum like PID controllers, Fuzzy interference system, Neuro-fuzzy interference system and Adaptive Neuro-fuzzy interference system.

In 2005, Ye Zhang build an Adaptive Neuro-fuzzy interference system controller that to balance an inverted pendulum using cart and pole, But he did not succeed in applying adaptive Neuro-fuzzy on real inverted pendulum hardware [1]. In 2006, Kok jiann Horng continued the incomplete project by Ye Zhang and tried to close the inverted pendulum hardware, but he did not succeed in closing the inverted pendulum hardware [2]. Yamakawa designed a high speed fuzzy controller hardware system and used only seven fuzzy rules to control the angle of an inverted pendulum [3].

1.2 Project aims and objectives

The overall aim of the project is to design, build inverted pendulum hardware and develop a controller using FIS algorithm to balance an inverted pendulum system in non-linear or multi-variable state.

- To achieve this aim, the following objectives must be accomplished
- Do research and understand the basic components required by the inverted pendulum system
- Study Zilog Z8 Encore!® Z8F64200100KIT MCU development board manual, study the specification of Z8 Encore!® MCU and understand how to configure Z8 Encore!® for this particular project
- Design and build the inverted pendulum hardware and assemble all the components
- Study and understand the fuzzy interference system (FIS), Learn how to create membership functions, rules and construct the FIS controller under MATLAB using FIS editor

University of Hertfordshire

- Learn C language and try to interface between motor and Incremental encoder by writing the program in ZDS II software (C compiler provided by ZILOG Z8 Encore!® Z8F64200100KIT MCU software)
- Do simulation in MATLAB to achieve a successful FIS controller
- Apply the FIS controller to the inverted pendulum hardware

1.3 overview of the project

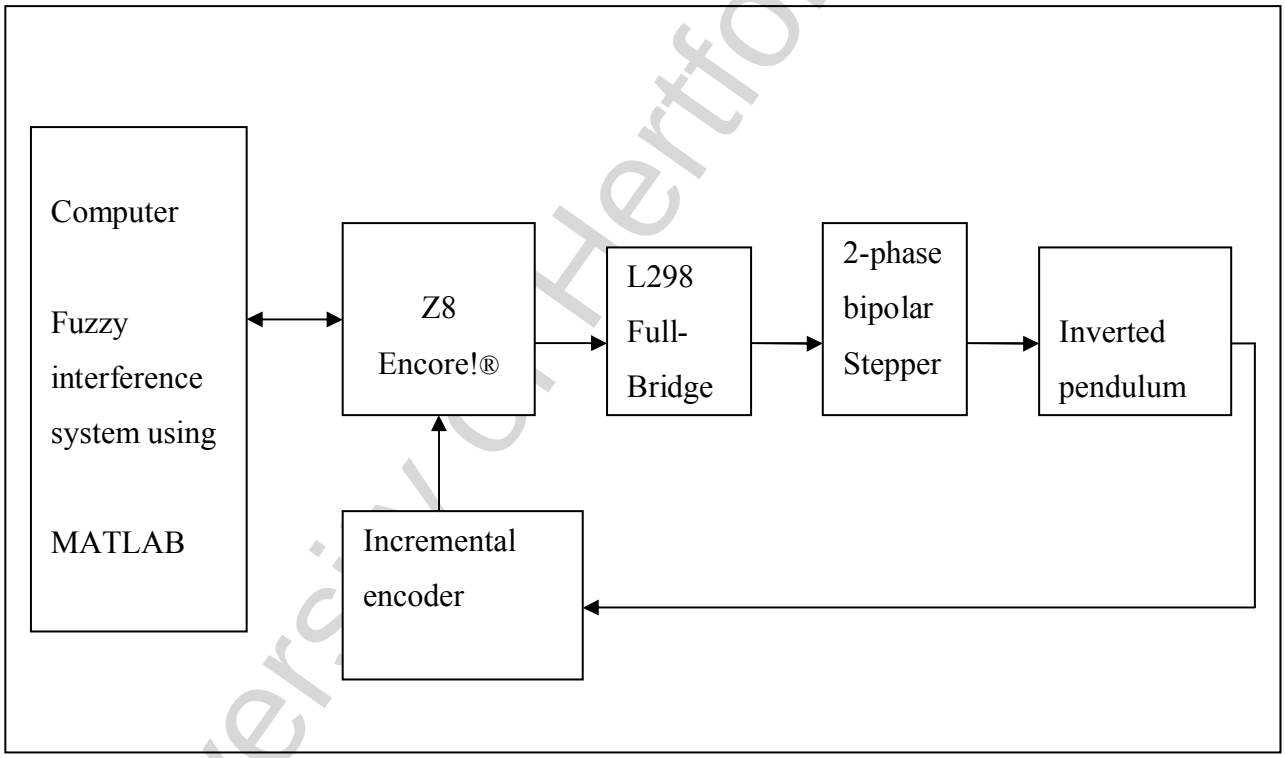


Figure 1.3 Inverted Pendulum Block Diagram

Inverted pendulum is an application of Mechatronics. Mechatronics means combination of different department (mechanical engineering, electronic engineering, computer engineering, control system and electrical engineering). Balancing of an inverted pendulum is a real-time controlling. Z8 Encore!® Z8F64200100KIT MCU is used to interface between software and the hardware. Pole angle and position will be measured using three channel optical incremental encoder module; output of the incremental encoder module is three waveforms (Channel A, Channel B and index Pulse). This generated output waveforms are send to the Z8

University of Hertfordshire

Encore!® Z8F64200100KIT MCU controller. Z8 Encore!® controller looks which channel is leading and which channel is lagging based on that it will generate a write sequence of 8-binary data on its port for energizing the motor windings. A dual Full-bridge driver is used to change the direction of the stepper motor. Figure shows the block diagram of the system.

1.4 Organisation of report

This report consists of 6 chapters listed below:

Chapter 1: Introduction

This chapter gives an overview of the project, project aims, objectives and a brief format of the report organisation is presented.

Chapter 2: Inverted Pendulum Hardware

In this chapter, all the major components of the inverted pendulum hardware will be discussed in detail. Design the frame work of the inverted pendulum and assemble them into a whole hardware structure. What are the specifications of the each component or parts?

Chapter 3: Programming and configuration of Z8 controller

In this chapter, discussions will be on Z8 controller programming and configuration. Assign of GPIO Pins for data input and output. Set up of interrupts and timer are outlined in detail

Chapter 4: FIS controller design

Design Fuzzy interference controller system. Analyse the inverted pendulum hardware and build the fuzzy set, membership functions and fuzzy rule to balance the inverted pendulum

Chapter 5: Implementation and testing

In this chapter, the result of the output incremental encoder and how the Z8 controller output is changing based on the output values of the encoder Results of the close loop inverted pendulum is shown in this chapter

Chapter 6: Power management

In this chapter, the time management, budget management and resources for the inverted pendulum is discussed

Chapter 7: Conclusion and further discussions

This is the final chapter of the project report. This chapter includes project conclusion, summarizing all the important findings and achievements of this project and comments. Potential further work for this project is highlighted.

University of Hertfordshire

2. Inverted Pendulum Hardware

In this chapter, all the major components of the inverted pendulum hardware are discussed in detail. The major parts or components of the inverted pendulum hardware are:

- Pole (second link), Shaft and First Link
- Motor
- Dual full-bridge L298 motor driving IC
- Opto-Coupler
- HEDS-9140 Incremental Encoder
- HEDS-5140 Code Wheel

2.1 Pole, Shaft and First Link

The First link on which Shaft is fixed is made of aluminium has a weight of 0.1Kg. This link is mounted on the 2 phase bipolar stepper motor shaft and tightened with a screw. The shaft to which the pendulum is attached is made of steel has a mass of 0.05Kg firmly fixed on the first link, able to rotate freely but no axial movement with less friction. Therefore two bearings are used to hold the shaft substantially with the first link. The pole on the shaft is 25cm in length and 0.03Kg in mass approximately; it is made of aluminium. The one end of the pole is fixed on the shaft, which gives the pole a 2 degree freedom of rotation (either Left or Right) rotates freely in vertical plane.

Furthermore, there must be no movement between the shaft and the pole, because the pole angle, position and direction of the pendulum is measured by the incremental encoder whose code wheel is fixed on other end of the shaft. Therefore two circular rings with nut is used to firmly fixed pole on the shaft. Figure 2.1 shows the diagram of the pole and the shaft (see Appendix D for Mechanical Drawings).

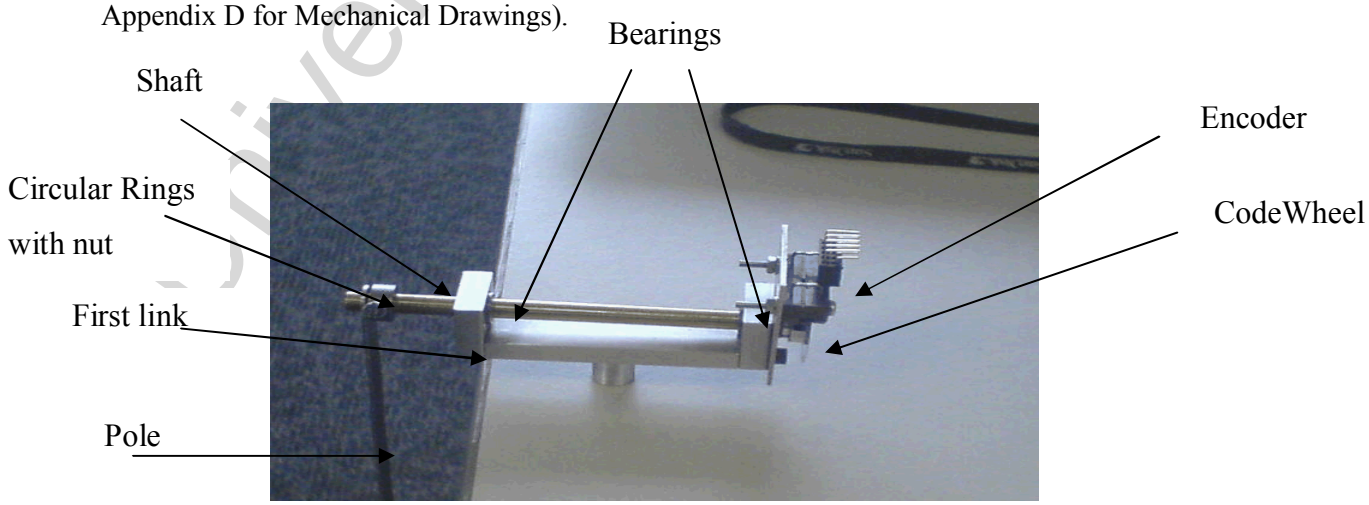


Figure 2.1 Pole, Shaft and First Link

University of Hertfordshire

2.2 Two-Phase Bipolar Stepper Motor:

2-phase bipolar stepper motor is the source of energy for the first link. From the data sheet [4]. 2-phase bipolar stepper motor has the following characteristics (See Appendix B for Crouzet stepper motor data sheet)

- Voltage at motor terminal : 6.6V
- Current per phase : 0.75A
- Resistance per phase : 9Ω
- Step Angle : 7.5°
- Absorbed Power : 10W
- Holding Torque : 180mN.m

Figure 2.2.1 Stepper Motor[4]

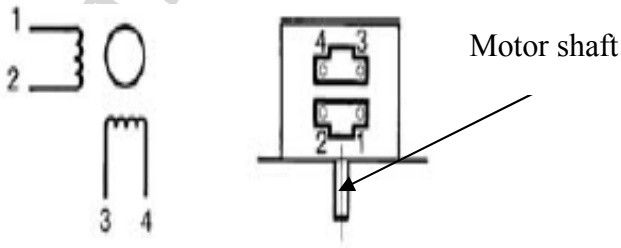


Figure 2.2.4 stepper motor connection

2-phase bipolar Stepper motor either rotates in clockwise or in anti-clockwise based on the command; that is energizing the various stepper motor coils in a particular sequence of pattern. Each pattern causes the stepper motor to move in one step. The wiring connections for the 2-phase bipolar stepper is shown in table 2.2.2 for clockwise and 2.2.3 for anti-clockwise rotation.

Step	1	2	3	4
1	-	+	-	+
2	-	+	+	-
3	+	-	+	-
4	+	-	-	+

Table2.2.3 Energizing Sequence

Step	1	2	3	4
1	+	-	-	+
2	+	-	+	-
3	-	+	+	-
4	-	+	-	+

Table 2.2.4 Energizing sequence

University of Hertfordshire

For Clockwise For anti-clockwise

The above table's 2.2.3 and 2.2.4 shows the configuration of each of the four wires numbering from 1 to 4; and the two middle wires are Center-tapped as shown in figure 2.2.2.

2.3 L298 Full-Bridge driver from National Semiconductor

L298 Dual Full-Bridge driver is a two H-bridge (back to back connected) IC used in this project to change the direction of the 2-phase bipolar stepper motor. Direction of the stepper motor is decided based on how the wirings or the terminals of the 2-phase stepper motor are energized. From the data sheet [5] L298 Full-Bridge driver has the following characteristics: (See Appendix C for electrical characteristics)

- Operating supply voltage up to 46V
- Total DC current up to 4A
- TTL compatible inputs
- Over temperature protection
- Low saturation voltage
- Logical 0 input voltage up to 1.5V (High noise immunity)

Based on the information from the date sheet [5], we briefly look into the function of the pins of the L298 Full-Bridge Driver IC. The pin layout of the L298 are shown in figure 2.3.1

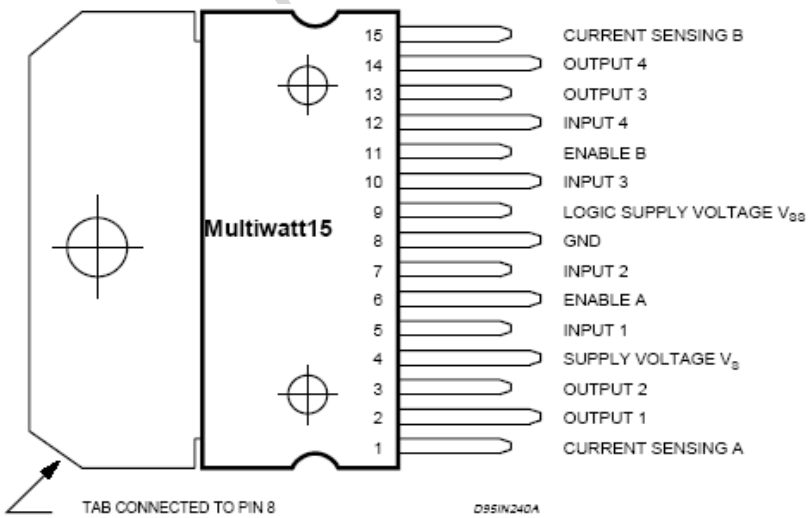


Figure 2.3.1 Pin Connections of the L28 Full-Bridge IC

Pin Functions used in the project:

1. Vs: Supply voltage for the power output stages.
2. Input1and Input2: TTL compatible inputs of the bridge A

3. Input3 and input4: TTL compatible inputs of the bridge B
4. Output1 and Output2: Outputs of the bridge A
5. Outputs3 and Outputs4: Outputs of the bridge B
6. GND: Ground
7. SenseA and SenseB: Between this pin and ground is connected the sense resistor to control the current of the load.
8. Enable A and Enable B: TTL compatible Enable input
9. VSS: Supply voltage for the logic Blocks. A 100nF capacitor must be connected between the pin and the ground.

Figure 2.3.2 shows the connection between the L298 Full-bridge driver IC, Z8 controller and the 2-phase bipolar stepper motor. PA0, PA1, PA2 and PA3 are the outputs from the Z8 controller of PortA are connected to the L298 full-bridge driver Input1, Input2, Input3 and Input4 respectively. The four wiring of the 2-phase bipolar stepper motor is connected to the L298 Full-bridge driver Outputs (output1, output2 …. output4) respectively. The L298 Full-bridge driver IC is powered by a separate 12V power supply.

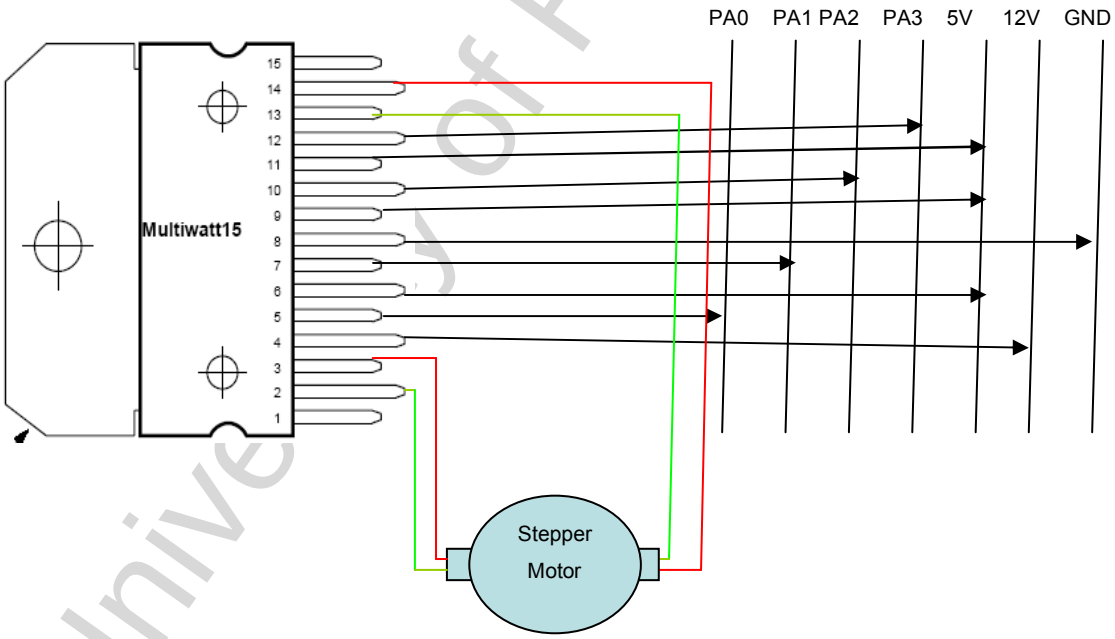


Figure 2.3.2 Connection between L298 IC, Z8 Controller and Motor

Diodes D1, D2….D8 are used to protect the L298 full-bridge driver IC. Because in inductive loads current flow from load to source when a motor accelerates or decelerates for any reason. These reverse current can damage the L298 full-bridge driver IC. Hence 2A fast

University of Hertfordshire

diodes are used to protect the L298 IC from reverse current coming from the load to source by turning ON when the reverse current exceeds it limit.

L298 integrates two power output stages. The power output stage is a bridge configuration and its output is used to drive a 2-phase bipolar stepper motor depending on the states of the input. Two Enable (Enable A and Enable B) inputs are provided to enable or disable the device independently of the input signals. If input1 is high and input2 is low then the current flow from output1 to output2 and if input1 is low and input2 is high then current flow from output2 to output1 respectively. Similarly if input3 is high and input4 is low then current flow from output3 to output4 and if input3 is low and input 4 is high then current flow from output4 to output3 respectively as shown in figure 2.3.2.

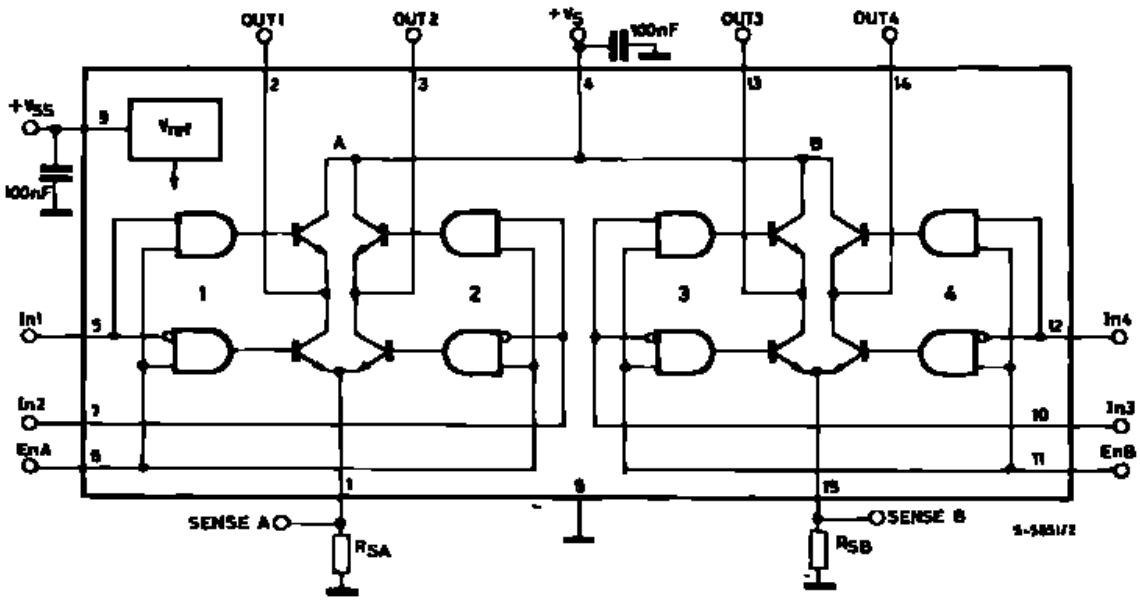


Figure 2.3.2 Block Diagram of L298 Dual Full-Bridge Driver []

The current, that flows through the load comes out from the bridge at the sense output: an external resistor (RS_A ; RS_B) used to detect the intensity of this current. Hence it is very important to connect sense resistors (RS_A ; RS_B) of 0.5Ω to pins 1 and 15 to a ground as shown in figure 2.3.2 above.

Each bridge is driven by four gates the input of which are In1;In2:EnA and In3;In4:EnB. Therefore EnA and EnB pins are connected to the 5V power supply. Vss supply voltage for the logic blocks is connected to the 5v supply. To drive 2-phase bipolar stepper

motor we require two H-bridges therefore enable both Enable inputs (EnableA and EnableB). Otherwise the L298 Dual Full-Bridge IC will not drive the 2 phase bipolar stepper motor.

Problem encountered and solution:

The voltage on the Z8 controller pins are 3.3V but the minimum voltage required by the L298 full bridge driver is 4.5V. Hence optocoupler is used to increase the voltage.

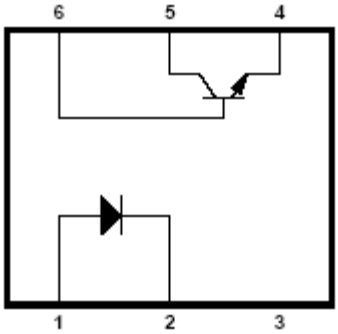


Figure 2.3.3 Functional Diagram

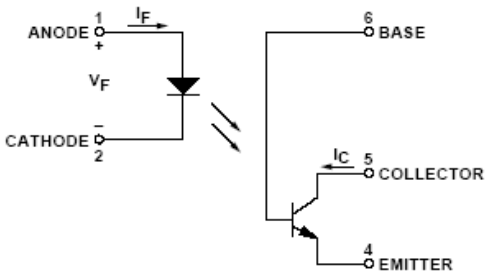


Figure 2.3.4 Schematic Diagram

- Anode
- Cathode
- Not connected
- Emitter
- Collector
- Base

4N25 Optocoupler used in this project. It contains a light emitting diode optically coupled to a photo-transistor as shown in above figure 2.3.4. From the data sheet [] it has the following characteristics:

Feature:

1. Current transform ratio (CTR) min 20% at I_F = 10 mA, V_{CE} = 10V
2. input-output isolation voltage (V_{iso} =2500 V_{rms})
3. Response time (tr:typ., 3µs at V_{CE} = 10V, I_C = 2mA, R_L = 100Ω)

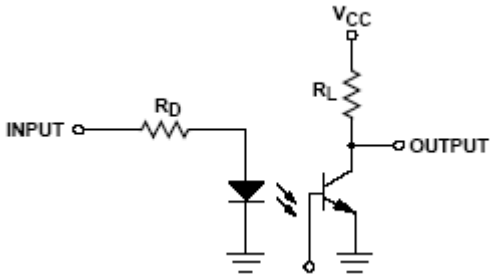


figure 2.3.5 Test circuit for time response

I have taken four opto-couplers for four Z8 controller output pins. Connection for the Opto-coupler is shown in above figure 2.3.5. Output from the Z8 controller is connected to the anode terminal and all cathodes are connected to the Z8 controller GND Pin number 39. A 5V power supply is applied across Vcc terminal, all base terminals are grounded and output from the emitter terminal is connected to the L298 Full-Bridge driver.

When I am applying the rated supply voltage required by the stepper motor that is 12V to the L298 Full-Bridge Driver IC, the L298 IC is getting heated up which can damage the L298 Dual full-bridge driver IC very easily. Therefore a heat sink is attached to L298 Full-Bridge Driver IC which absorbs the heat dissipated by the L298 IC. Hence these avoid any damages or burns to L298 driver IC.

2.4 HEDS-9140 Incremental Encoder from Hewlett-Packard

In inverted pendulum, the measurement of pole angle, position and direction of rotation with respect to the upright (absolute zero) position is very important. If the accuracy of the system is good (that is if we get the precise values of pole angle and position) then the system is more the robust. Several sensors like potentiometer, shaft encoders and absolute encoders are available in market. To avoid any friction on the shaft incremental encoders or the absolute encoders are the right choice because these encoders avoids any mechanical contacts required to know the pole angle, position or the direction of the pole. But in the market absolute encoders are too expensive compare to incremental encoders. After looking so many encoders and the price of the encoders hence I decided to go for the following encoder.

- HEDS-9140 three channel optical incremental encoder
- HEDS-5140 Code wheel

2.4.1 HEDS-9140 has the following characteristics listed below from the data sheet [6] (See Appendix D for HEDS-9140 characteristics)

- Two Channel Quadrature output with Index Pulse
- Resolution up to 2000 CPR (counts per revolution)
- Low Cost
- Easy to mount
- No signal Adjustment required
- Small size
- -40°C to 100°C Operating Temperature

University of Hertfordshire

- TTL Compatible
- Single 5V supply voltage

HEDS-9140 has two channel quadrature outputs (waveforms out of phase) in addition with a third channel index output. This index output is a 90 electrical degree high true index pulse which is generated once for each full rotational of the code wheel. HEDS-9140 is designed for use with a HEDS-5140 code wheel which has an optical radius of 11.00mm (0.433) (from data sheet).

HEDS-9140 is an emitter or detector module. Coupled with a code wheel, this module translates the rotary motion of a shaft into a three channel digital output.

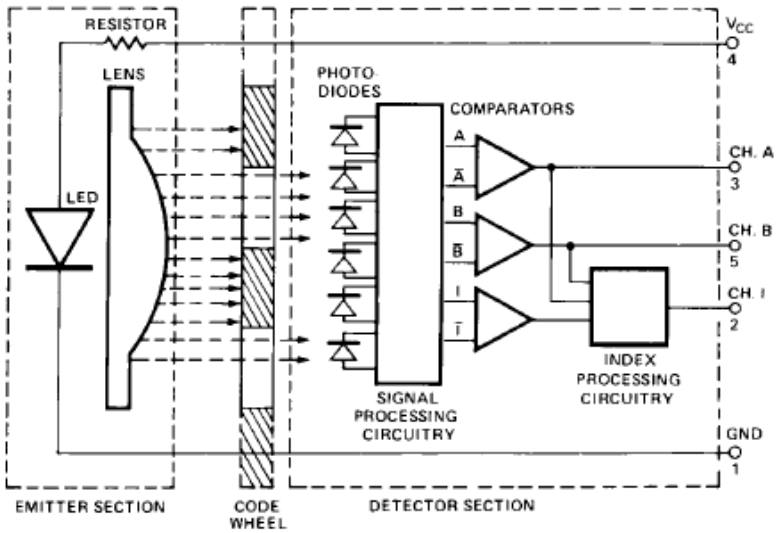


Figure 2.4.1.1 HEDS-9140 Block Diagram [6]

If we look at the figure2.4.1.1 block diagram, HEDS-9140 contains a single light emitting diode (LED) as its light source. The light collimated into a parallel beam by means of a single polycarbonate lens located directly over the LED. Opposite the emitter is the integrated detector circuit. This IC consists of multiple sets of photo detectors and the signal processing circuitry necessary to produce the digital waveforms (from data sheet).

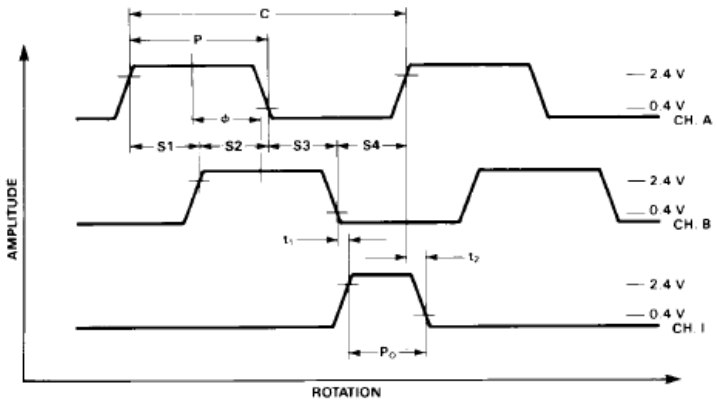


Figure 2.4.1.2 HEDS-9140 Output Waveform [6]

The code wheel rotates between the emitter and detector, causing the light beam to be interrupted by the pattern of spaces and the bars on the code wheel. Photo diodes which detect this interrupt are arranged in a pattern that corresponds to the radius and design of the code wheel. There detectors are placed in such a way that a light period on one pair of detectors corresponds to a dark period on the adjacent pair of the detectors. Because of this integrating phasing technique, the digital output of channel A is in quadrature with channel B(90° degree out of phase) as shown in figure below 2.4.1.2 (from data sheet).

If the code wheel rotates in the direction of the arrow on the top of the module then Channel A will lead the Channel B by 90°. If the code wheel rotates in the other direction that is opposite to the arrow on the top of the module then Channel B will lead the Channel A by 90°. As shown in below figure 2.4.1.3

University of Hertfordshire

Figure 2.4.1.4 HEDS-9140 Channel A(1>) and channel B (2>) Output waveform from Oscilloscope

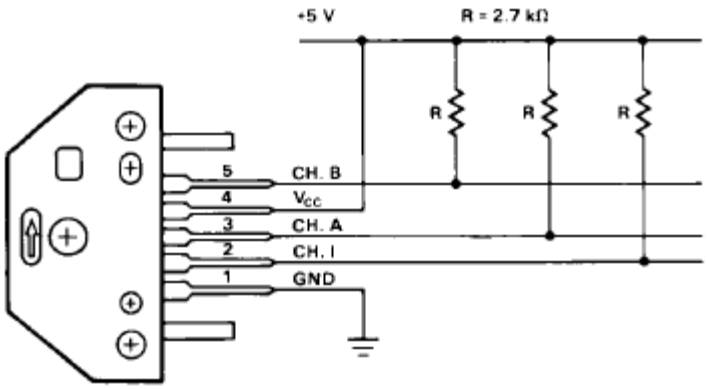


Figure 2.4.1.4 Pull –up resistors on HEDS-9140 encoder output [6]

To ensure reliable encoding performance, HEDS-9140 three channel optical incremental encoder module require 2.7KΩ pull-up resistors on output pins 2, 3 and 5 that is on Channel A, Channel B and channel index output as shown in above figure 2.4.1.4.

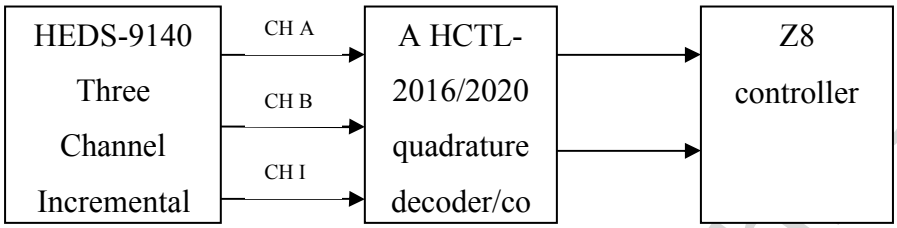


Figure 2.4.1.5 Diagrams show typical interface between Z8 controller and the channel optical Incremental Encoder module

Problem Encountered and Solution:

A HCTL-2016/2020 quadrature decoder/counter is not compatible to Z8 controller it requires external clock with higher frequency than the Z8 controller which is quite difficult to achieve using the available equipments in the lab. Hence this problem can be solved via software.

In year 2006 Kok Jiann Horng tried to interface between Z8 controller and incremental encoder via hardware approach, that is using HCTL-2016/2020 quadrature decoder/counter but he is fail to close the inverted pendulum hardware. Hence this problem can be solved through software approach by using timers. In Z8 controller we have counter option by using this we can count the number of pulses. Hence there is no need to go for decoder/counter circuit.

2.4.2 HEDS-5140 Code wheel

As mentioned earlier code wheel rotates between the emitter and detector, causing the light beam to be interrupted by the pattern of spaces and bars on the code wheel. The photodiodes which detects this interrupts are arranged in a pattern that corresponds to the radius and design of the code wheel. From the data sheet [7] HEDS-5140 two and three channel code wheel optical encoder mode has the following characteristics:

1. Resolutions from 96 CPR (counts per revolution) to 2048 CPR
2. Code wheel available in glass, film and metal

University of Hertfordshire

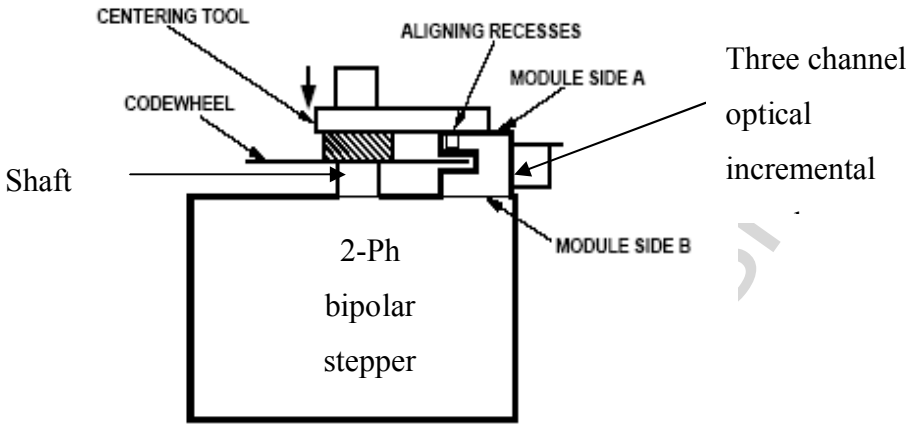


Figure 2.4.2.1 Alignment tool is used to set height of the code wheel

Assembly tools required for alignment Code wheel on HEDS-9140 Incremental encoder is for Centering and Gap-Setting HEDS-8905 alignment tool is required shown in figure 2.4.2.1.

University of Hertfordshire

3. Programming and configuration of the Z8 controller

In this chapter, we will see how the Zilog Z8 Encore!® Z8F64200100KIT MCU is programmed and configured to control the inverted pendulum hardware.

3.1 Overview and Configuration of the Zilog Z8 Encore!® ZF642X MCU controller

Specifications and main features of the Zilog Z8 Encore!® Z8F64200100KIT MCU controller are as follows (from Z8 Encore! User Manual)[8].

- 20MHz eZ8 CPU
- Up to 64KB Flash Memory with in circuit programming capability
- Up to 4KB register RAM.
- Seven 8-bit (Ports A-G) and one 4-bit (Port H) general purpose input/output (GPIO) pins
- 12-channel, 10-bit analog-to-digital converter(ADC)
- Two full-duplex 9-bit UARTs with bus transceiver Driver Enable Control
- I2C
- Serial Peripheral Interface
- Two Infrared Data Association (IrDA) compliant infrared encoder/decoders
- Up to four 16-bit timers with capture, compare, counter and PWM capability
- Watch-Dog timer (WDT) with internal RC oscillator
- 3-channel DMA
- 24 interrupts with configuration priority
- On-chip Debugger
- Power-On reset (POR)
- 3.0-3.6V operating voltage with 5V-tolerant inputs

Below figure 3.1.1 shows a diagram of the Z8 Encore!® Z8F64200100KIT MCU controller r (For convenience purpose Zilog Z8 Encore!® Z8F64200100KIT MCU is know called as Z8 controller in this project report). Among the many features, only certain features of the Z8 controller are used in this project such as GPIO ports, Interrupts and Timers.

University of Hertfordshire

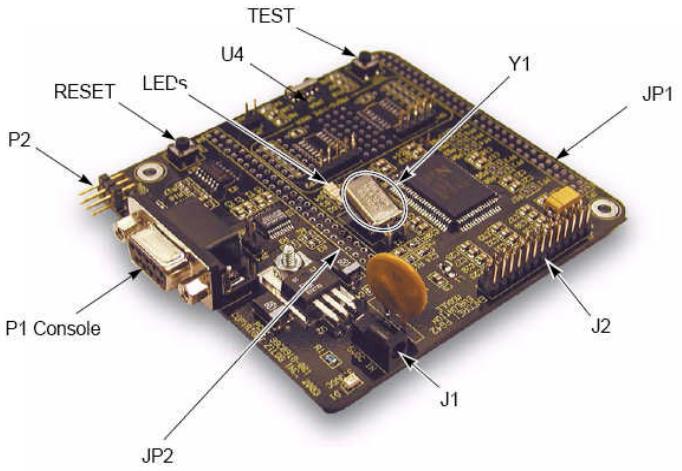


Figure 3.1.1 Z8 Encore!® 64K serial MCU development board [8]

In this project, Z8 controller works as a interface between the PC (software programming) and the hardware (inverted pendulum). The Z8 controller read the signals from the incremental encoder on one of the GPIO ports and sends them to PC as input data, and then according to the output data from PC, the Z8 controller will generate the PWM signal for the 2-ph bipolar stepper motor.

3.2 Configuration of general purpose input/output (GPIO)

As mentioned earlier, the 64K series Z8 controller having seven 8-bit ports (Ports A-G) and one 4-bit port (Port H) for general purpose input/output (GPIO). Each pin is individually programmable for input or output operations. Some bits of the ports can be configured for special functions, such as interrupts, timers and UARTs etc.

In this project, two out of the eight ports are used. 4-pins of the PortA are configured as output ports (PA0, PA1, PA2 and PA3) and 2-pins of the PortD are configured as input ports (P3AD and P5AD). The first 4-pins of the port A is connected to the L298 full-bridge driver to control the direction and current applied to the 2-phase bipolar stepper motor and PortD pin P3AD is configured for special function (that is for interrupt purpose); an interrupt occurs when ever P3AD goes in high state the interrupt request allow device to suspend CPU operation and for CPU to start an interrupt service routine and hence an interrupt service routine code is executed during this period. 2-pins of PortD are connected to the incremental encoder output Channel A and channel B.

University of Hertfordshire

Four registers for each port provide access to GPIO control, input data, and output data. Port A-H Address and Control registers are used together to provide access to sub-registers for port configuration and control (from user manual)
Replace "x" with A-H accordingly

Port Register Mnemonic	Port Register Name
PxADDR	Port A-H sub-registers(select sub-registers)
PxCTL	Port A-H Control Register (Provides access to sub-registers)
PxIN	Port A-H Input Data Register
PxOUT	Port A-H Output Data Register
Port sub-Register Mnemonic	Port Register Name
PxDD	Data Direction
PxAF	Alternate Function
PxOC	Output Control (open-drain)
PxDD	High Drive Enable
PxSMRE	STOP Mode Recovery Source Enable

Table 3.2.1 GPIO Port Registers and Sub-Registers

1. PxADDR is the address register used to select port sub-registers. Three port sub-registers that are used in this project are
   - ALT_FUN: Alternate function sub-register. Used to configure a certain bits of a certain port as alternative function.
   - DATA_DIR: Data direction sub-register. Used to configure port pins as input or output
   - OUT_CTL: Output control sub-register. Used to enable or disable the port drain.
2. PxCTL is the control register; if the address register is used to select a sub-register then the control register is used to configure a sub-register.
3. PxIN is the input register; 8-bit input data is stored in this register when a GPIO port is set as input. Input data register are Read-Only.
4. PxOUT is the output register, 8-bit output data is stored in this register before sending out from output GPIO ports.

University of Hertfordshire

In my project I have chosen Port A as output. Configure PortA as output,

Alternate Functions:

I am not using any alternate functions of PortA like timers, UARTs or I2C of PortA. So the 8-bit configuration value for the control register will be 00000000 = 0x00H.

PAADDR = ALT_FUN;

PACTL = 0x00;

Data Direction:

BITS	7	6	5	4	3	2	1	0
FIELD	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
RESET	1							
R/W	R/W							
ADDR	If 01H in Port A-H Address Register, accessible through Port A-H Control Register							

Table 3.2.1 Data Direction Sub-Registers [8]

0 = Output. Data in the PortA Output Data Register is driven onto the port pin.

1 = Input. The port pin is sampled and the value written into the portA Input Data Register. The Output is tri-stated.

I am configuring PortA as Output, so the 8-bit configuration value for the control register will be 00000000 = 0x00H

PAADDR = DATA_DIR;

PACTL = 0x00;

Output Control:

BITS	7	6	5	4	3	2	1	0
FIELD	POC7	POC6	POC5	POC4	POC3	POC2	POC1	POC0
RESET	0							
R/W	R/W							
ADDR	If 03H in Port A-H Address Register, accessible through Port A-H Control Register							

Table 3.2.2 Port Output Control Sub-Registers [8]

0 = drain enable

1 = drain disable

Here drain values for all the ports are enabled. So, the 8-bit configuration value for control register will be 00000000 = 0x00H. Otherwise high value is not driven if the drain has been disabled by setting the corresponding port output control register bit to 1.

PAADDR = OUT_CTL

PACTL = 0x00;

Port Output (PAOUT):

These bits contain the data to be driven out from the port pins. The values are only driven if the corresponding pin is configured as an output and the pin is not configured for alternate function operation.

BITS	7	6	5	4	3	2	1	0
FIELD	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0							
R/W	R/W							
ADDR	FD3H, FD7H, FDBH, FDFH, FE3H, FE7H, FEBH, FEFH							

Table 3.3.3 PortA Output data sub-registers [8]

POUTx = Port Output

0 = Drive a logical 0 (Low)

1 = Drive a logical 1(high). High value is not driven if the drain has been disabled by setting the corresponding port output control register bit to 1.

Where “x” indicates the specific GPIO Port pin number

In my project motor has 4 leads; hence the PAOUT is configured based on the energizing sequence of the 2-phase bipolar stepper motor winding.

I have chosen PA0, PA1, PA2 and PA3 as output pins of PortA.

If we want the 2-phase bipolar stepper motor to rotate in clockwise direction or anticlockwise, then the corresponding pin is configured as an output.

University of Hertfordshire

For Clockwise	For Anti-Clockwise
Case 1: First step PAOUT = 0x05 PAOUT = 00000101 Break; Case 2: Second step PAOUT = 0x06 PAOUT = 00000110 Break; Case 3: Third step PAOUT = 0x0A PAOUT = 00001010 Break; Case 4: Fourth step PAOUT = 0x09 PAOUT = 00001001 Break;	Case 1: First step PAOUT = 0x09 PAOUT = 00001001 Break; Case 2: Second step PAOUT = 0x0A PAOUT = 00001010 Break; Case 3: Third step PAOUT = 0x06 PAOUT = 00000110 Break; Case 4: Fourth step PAOUT = 0x05 PAOUT = 00000101 Break;

Table 3.2.4 Energizing sequence for clockwise and anticlockwise

I wrote this code in C programming and I have observed its output on the LEDS connected on the bread board in series with the RL = 100Ω resistance to avoid any damages to the LEDS. Hence the LEDS are blinking in right sequence for both clockwise and anticlockwise as shown in below figure 3.2.5.

University of Hertfordshire

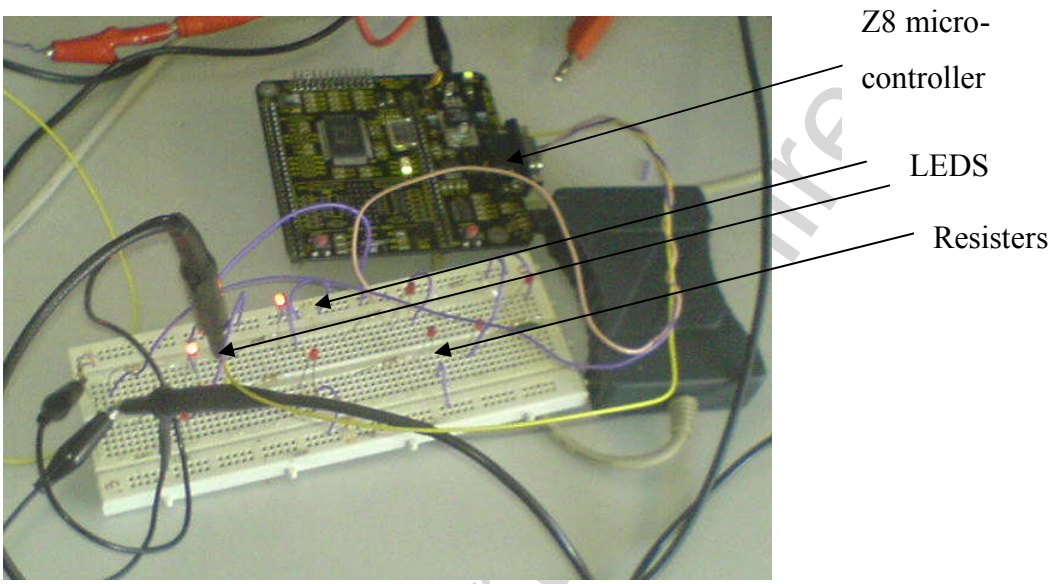


Figure 3.2.5 show the connection between Z8 controller and LEDS

After testing this, I have given this 4-pin output of PortA (PA0, PA1, PA2 and PA3) and GND from Z8 controller to the L298 Full-Bridge driver inputs (input1, input2, input3 and input4).

3.3 Configuration of Interrupt

Interrupt requests (IRQs) allow device to suspend CPU operation in an orderly manner and force the CPU to start an Interrupt Service Routine (ISR), usually interrupt service routine is involved with the exchange of data, control information and status information between the CPU and the interrupting device. When the Interrupt service routine is completed, CPU returns to the operation from which it was interrupted.

Interrupt controllers support three levels of interrupt priority shown in below figure 3.3.1, From (Zilog user manual) Level3 is the highest priority, Level2 is the second priority, and Level1 is the lowest priority. If all the priorities are enabled with identical interrupt priority then interrupt priority would be assigned from highest to lowest.

University of Hertfordshire

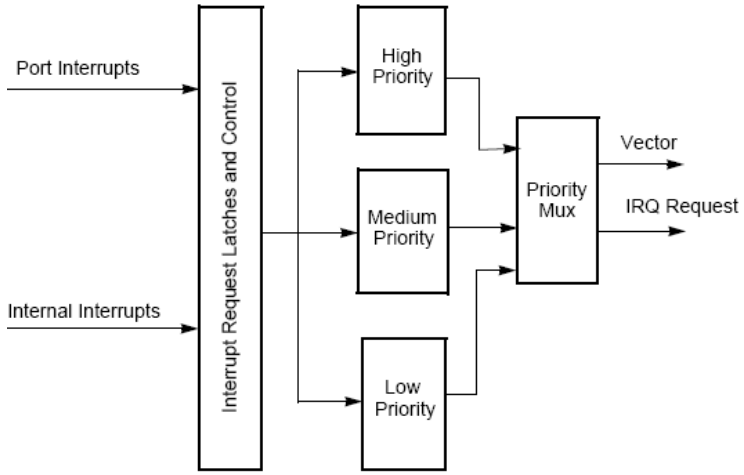


Figure 3.3.1 Interrupt controller Block Diagram [8]

In Z8 controller the ports support interrupts, they are PortA, PortC and PortD. I have chosen PortA for energizing 2-phase bipolar stepper motor. I have chosen PortD for interrupt in my project.

Configure PortD for interrupt:

2-bits of PortD are used to read the output signal from the incremental encoder channel and channel B. Here I have chosen Pin3 and pin5 of PortD as an input to the Z8 controller. Pin3 is connected to the incremental encoder output channel A and pin5 is connected to incremental encoder output channel B.

If code wheel rotate in the direction of the arrow of the encoder module then channel A will lead the channel B otherwise channel B will lead the channel A. One interrupt is enough to know which channel is leading and lagging.

3.3.1 Interrupt Port Initialization:

Void init_p3ad (void)

SET_VECTOR (P3AD, isr_p3ad)

The interrupt request 1 (IRQ1) register stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ1 register becomes 1 (From Zilog User Manual).

University of Hertfordshire

BITS	7	6	5	4	3	2	1	0
FIELD	PAD7I	PAD6I	PAD5I	PAD4I	PAD3I	PAD2I	PAD1I	PAD0I
RESET	0							
R/W	R/W							
ADDR	FC3H							

Table 3.3.2 Interrupt Request port initializatioin [8]

PADxI = Either PortA or PortD interrupt request

0 =No interrupt request is pending for GPIO PortA or PortD

1 = An interrupt request from GPIO PortA or PortD is awaiting service

Where "x" indicates the specific GPIO Port pin number.

Firstly, we have to initialize the PortD pin numbers 3 and 5.

I am not using any alternate function of PortD that is timers, UART or Watch-Dog timer. So the 8-bit configuration value of control register values 00000000 = 0x00H

```
PDADDR = ALT_FUN;
PDCTL = 0x00;
PDADDR = DATA_DIR;
PDCTL |= 0x28
```

0 = Output. Data in the PortD Output Data Register is driven onto the port pin.

1 = Input. The port pin is sampled and the value written into the portA Input Data Register. The Output is tri-stated.

Configure the data direction of 2-bits of PortD that is pin3 and pin5 by writing the desired configuration value to the control register. So the 8-bit configuration value for the control register will be 00101000 = 0x28H.

3.3.2 Set Interrupt priority:

The IRQ1 Enable High and Low bit registers form a priority. Priority is generated by setting bits in each register, How to set interrupt priority is shown in below table (From Zilog User Manual).

IRQ1ENH[x]	IRQ1ENL[x]	Priority	Description

University of Hertfordshire

0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level3	High

Table 3.3.3 Priority configuration table

Where x indicates the register bits from 0 through7

In my project interrupt is very important to know in which direction pendulum is rotating. Hence i have chosen high interrupt priority that is Level3 by setting both Enable High and Enable Low bit registers of P3AD of PortD to 1. So the 8-bit configuration value of Interrupt request priority is (IRQE) will be 00001000 = 0x08

IRQ1E0 |= 0x08;
IRQ1E1 |= 0x08;

3.3.3 Interrupt Edge selection Register

Interrupt edge selection register determines whether an interrupt is generated for the falling edge or rising edge on the selected GPIO port input pin.

BITS	7	6	5	4	3	2	1	0
FIELD	IES7	IES6	IES5	IES4	IES3	IES2	IES1	IES0
RESET	0							
R/W	R/W							
ADDR	FCDH							

Table 3.3.4 interrupt Edge Select Register [8]

IESx = Interrupt Edge Select x
0 = An interrupt request is generated on the falling edge of the PDx input.
1 = An interrupt request is generated on the rising edge of the PDx input
Where “x” indicates the specific GPIO Port pin number

University of Hertfordshire

Here I have chosen interrupt request on rising edge. So the 8-bit configuration value of Interrupt edge select register (IRQES) will be 00001000 = 0x08

IRQES |= 0x28;

3.3.4 Interrupt Port Select:

The interrupt port select register determines the port pin that generates interrupts. This register allows either PortA or PortD pins to be used as interrupts.

BITS	7	6	5	4	3	2	1	0
FIELD	PAD7S	PAD6S	PAD5S	PAD4S	PAD3S	PAD2S	PAD1S	PAD0S
RESET	0							
R/W	R/W							
ADDR	FCEH							

Table 3.3.5 interrupt Port Select Register [8]

PADxS = PAx/PDx Selection

0 = PAx is used for the interrupt for PAx/PDx interrupt request

1 = PDx is used for the interrupt for PAx/PDx interrupt request

Where “x” indicates the specific GPIO Port pin number

I have chosen PortD for Interrupt request. So the 8-bit configuration value for the interrupt select port register (IRQPS) will be 00101000 = 0x28H

IRQPS |= 0x28

3.4 Configuration of Timer

The internal Timer1 and Timer2 is used in this project to generate two Pulse Width modulation (PWM) externally using another Z8 controller (in place of encoder temporarily), because my hardware was not ready. There were so many disturbances in the engineering department because of school renovation. Therefore I generated the Pulse width modulation which is similar to the output of the three channel optical incremental encoder.

The Z8 controller has 4 internal timers they are Timer0, Timer1, Timer2 and Timer3. Timer1 and Timer 2 are configured as PWM mode, PWM signal are send from PC0 (PC0_T1OUT) and PC7 (PC7_T2OUT) to the main Z8 controller pins PD3 and PD5 to read the output signal

University of Hertfordshire

generated by the temporarily Z8 controller (Used in place of encoder). Each timer has 8-bit registers namely

- TxL = Low Byte Register
- TxH = High Byte Register
- TxRL = PWM Low Byte Register
- TxRH = PWM High Byte Register
- TxPWML = PWM Low Byte Register
- TxPWMH = PWM High Byte Register
- TxCTL0 = Timer Control 0 Register
- TxCTL1 = Timer Control 1 Register

Where "x" is the timer number (0 to3)

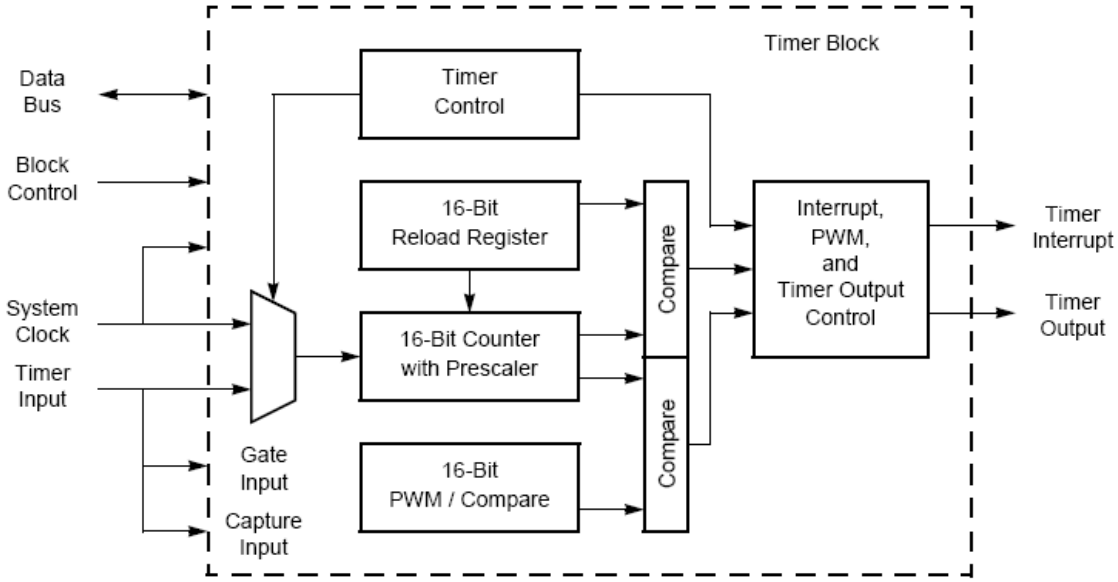


Figure 3.4.1 Timer Block Diagram [8]

The timer input is the system clock, the timers count up to 16 bit PWM match value stored in TxPWMH and TxPWML byte registers. When the timer count value matches the PWM value, the timer output toggles.

If TPOL (Timer input/output polarity) bit in the timer control 1 register is (From Zilog manual)

- Set to 1, the Timer Output signal begins as High (1) and then transitions to a Low (0) when the timer value matches the PWM value. The timer Output signal returns to a High (1) after the timer reaches the Reload value and is reset to 0001H.
- Set to 0, the Timer Output signal begins as Low (0) and then transitions to a High (1) when the timer value matches the PWM value. The timer Output signal returns to a High (1) after the timer reaches the Reload value and is reset to 0001H.

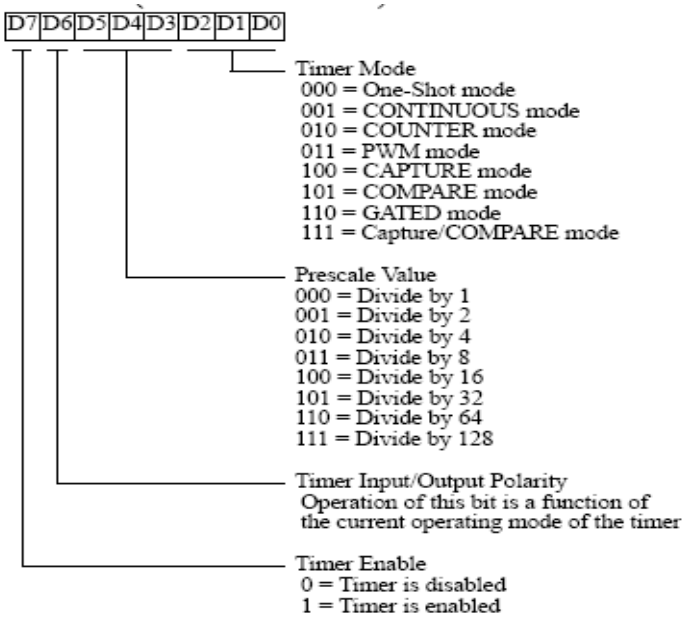

D7 D6 D5 D4 D3 D2 D1 D0
Timer Mode
000 = One-Shot mode
001 = CONTINUOUS mode
010 = COUNTER mode
011 = PWM mode
100 = CAPTURE mode
101 = COMPARE mode
110 = GATED mode
111 = Capture/COMPARE mode
Prescale Value
000 = Divide by 1
001 = Divide by 2
010 = Divide by 4
011 = Divide by 8
100 = Divide by 16
101 = Divide by 32
110 = Divide by 64
111 = Divide by 128
Timer Input/Output Polarity
Operation of this bit is a function of
the current operating mode of the timer
Timer Enable
0 = Timer is disabled
1 = Timer is enabled

Figure 3.4.2 Timer Control 1 Register [8]

Steps for configure a Timers for PWM mode (Zilog user Manual)

1. First Configure the 8-bit Timer control 1 register,
   - Disable the timer,
   - Set the TPOL,
   - Prescale value and
   - And configure the timer for PWM mode.

TxCTL1 = 01101101 =

2. Write the Timer High and Low Byte registers to set the starting count value.
3. Write to the PWM high and Low Byte registers to set the PWM value
4. Write to the Timer Reload High and Low Byte registers to set the Reload value. The Reload value must be greater than the PWM value
5. Configure the PortC pin for the Timer Output alternate function
6. Write the timer control 1 register to enable the timer.

The PWM period is calculated using the following equation:

$$\text{PWM period (s)} = \frac{\text{Reload value x Prescale value}}{\text{System Clock Frequency (Hz)}}$$

The TPOL is set to 1, the ratio of the PWM output High time to the total period is calculated by the following equation

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

For example if Reload value is FFFF, the prescale value is set to 32 and system clock frequency of Z8 controller is 18.432MHz, then the PWM period is

$$\text{PWM period} = \frac{65536 \times 32}{18432000}$$

$$= 0.1137 \text{ s}$$

$$= 113.7 \text{ ms}$$

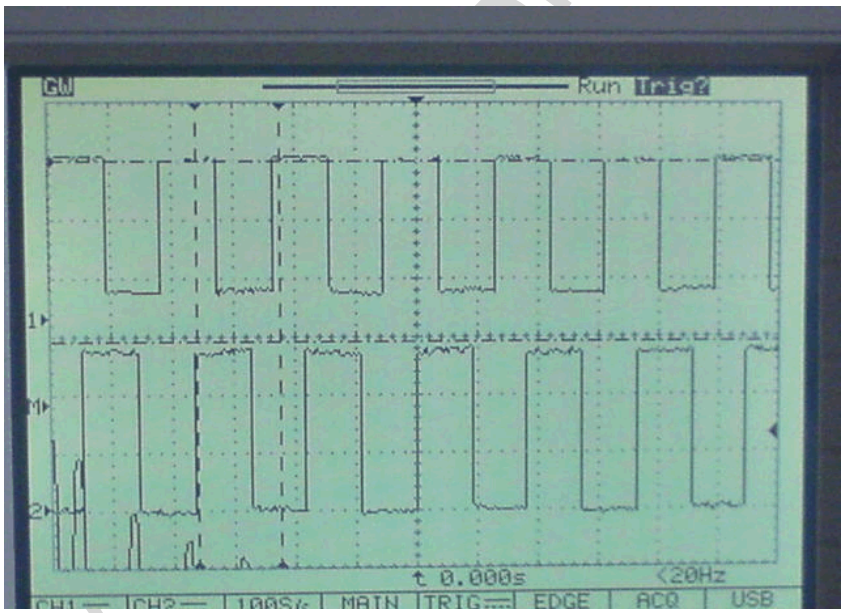


Figure 3.4.2 Timer1 (1>) and Timer (2>) Output waveform from oscilloscope

Two pulses are created similar to the encoder module output that is 90 degree out of phase by giving delay between two timers as shown in figure 3.4.2. The phase difference between two timers can be adjusted by setting delay time.

Therefore by changing the PWM value in the Timer PWM High (TxPWMH) and Low (TxPWML) Byte registers, we can change the duty cycle of the modulated PWM pulse.

University of Hertfordshire

4. Fuzzy interference system (FIS)

In this chapter, discussion will be on Fuzzy Interference controller, used to control the inverted pendulum hardware. Discussions will be on creating fuzzy sets, membership function, fuzzy rules and design considerations.

4.1 FIS introduction

Fuzzy interference system is the process of mapping between given inputs to an output, using the theory of fuzzy logic. Fuzzy logic mimics the human decision making by using fuzzy rules with vague terms. It represents expert knowledge with fuzzy sets

The concept of fuzzy logic was proposed by Lofti Zadeh, A professor at the University of California at Berkley. Fuzzy logic can be implemented on hardware (inverted pendulum), software (simulation) or on both. Fuzzy logic is simplest way of mapping between input and output.

The principle of fuzzy inference system is a list of if-then statements called rules. Before we build a system that interprets rules we have to define the linguistic variables (pole angle, angular velocity and torque) and linguistic values (low, medium, and high) called membership function and fuzzy sets.

A fuzzy set can be simply defined as a set with fuzzy boundaries [13]

Let X be the universe of discourse and its elements be denoted as x. in classical theory, crisp set A of X is defined by a function $f_A(x)$ called the characteristic function of A [13].

$$f_A(x) : X \rightarrow (0,1)$$

Where,

$$f_A(x) = \begin{cases} 1, \text{if } x \in A; \\ 0, \text{if } x \notin A. \end{cases}$$

This function map universe of discourse X to a two elements. For any elements of x of universe X, $f_A(x)$ (characteristic function) is equal to 1 if x is an element of set A and equal to zero if x is not an element of set A.

In fuzzy theory, fuzzy set A of universe X is defined by function $\mu_A(x)$ called membership function of set A [13].

$$\mu_A(x): X \rightarrow (0,1)$$

Where,

$\mu_A(x) = 1$ if x is totally in A;
$\mu_A(x) = 0$ if x is not in A;
$0 < \mu_A(x) < 1$ if x is partly in A.

University of Hertfordshire

4.2 Design of fuzzy controller

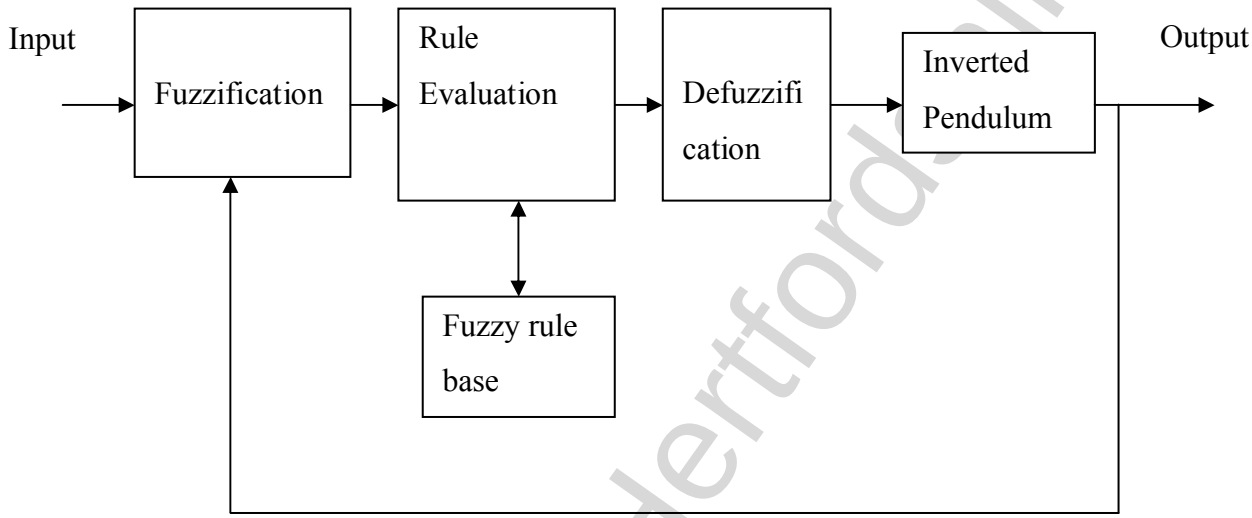


Figure 4.2.1 Block diagram of Fuzzy logic controller

Fuzzy interference system is basically divided into three steps they are

- Fuzzification
- Rule evaluation and
- Defuzzification

4.2.1 Fuzzification

Fuzzifier takes the input variables based on that it will determine the fuzzy set and membership functions. Here the input to the fuzzifier is pole angle, angular velocity and output from the fuzzy controller is the torque. The inverted pendulum can rotate in two directions either left or right. Hence define the pole angle and angular velocity by defining the membership function for the fuzzy sets

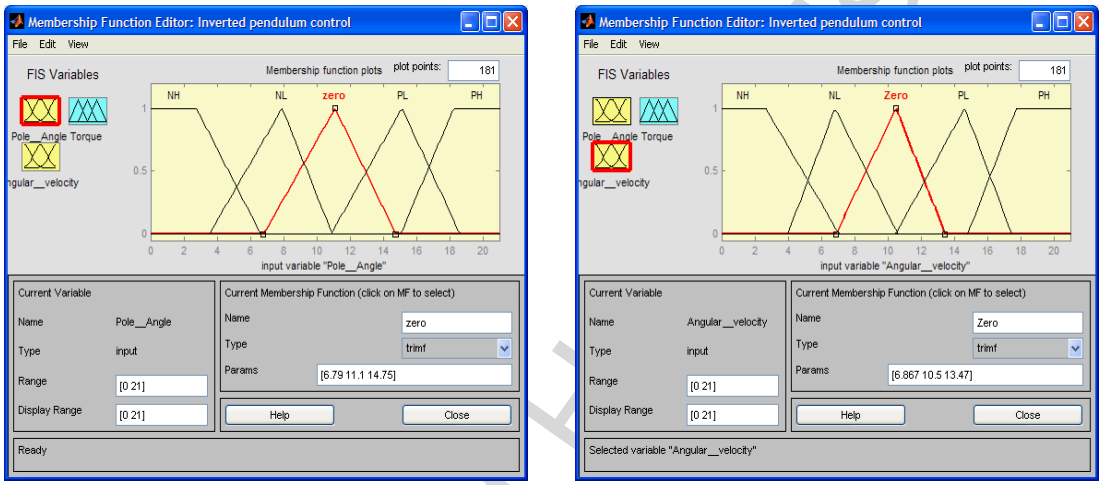
- Negative high (NH)
- Negative low (NL)
- Zero (Zero)
- Positive low (PL)
- Positive high (PH)

University of Hertfordshire

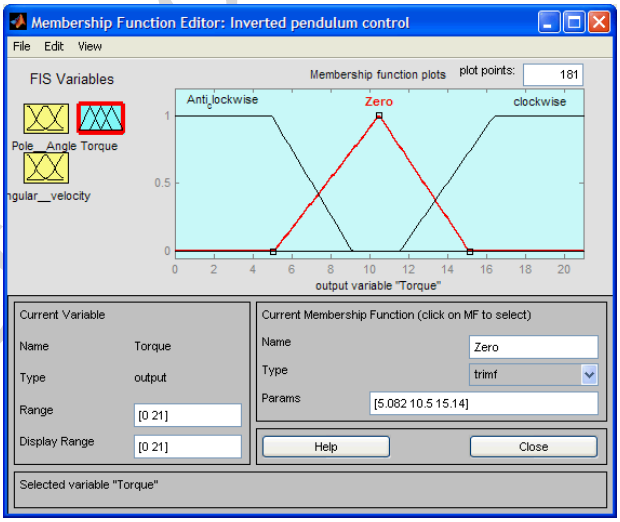
Similarly for toque

- Anti-Clockwise (AL)
- Zero (Zero)
- Clockwise (CL)

The fuzzy set and membership function for the input and output variables are shown figure 4.2.1.1.



a) Input membership functions



b) Output membership functions

Figure 4.2.1.1 Memebership functions a) Input membership functions and b) output membership fucntions

4.2.2 Rule evaluation

In this stage fuzzy interference system takes the fuzzified inputs and applies them to the fuzzy rule base. If a fuzzy rule has multiple rules then fuzzy operator (AND or OR) is used to create a single number which represents the truth value of the fuzzy rule. Some rules for the inverted pendulum is created and shown in below figure using MATLAB FIS editor.

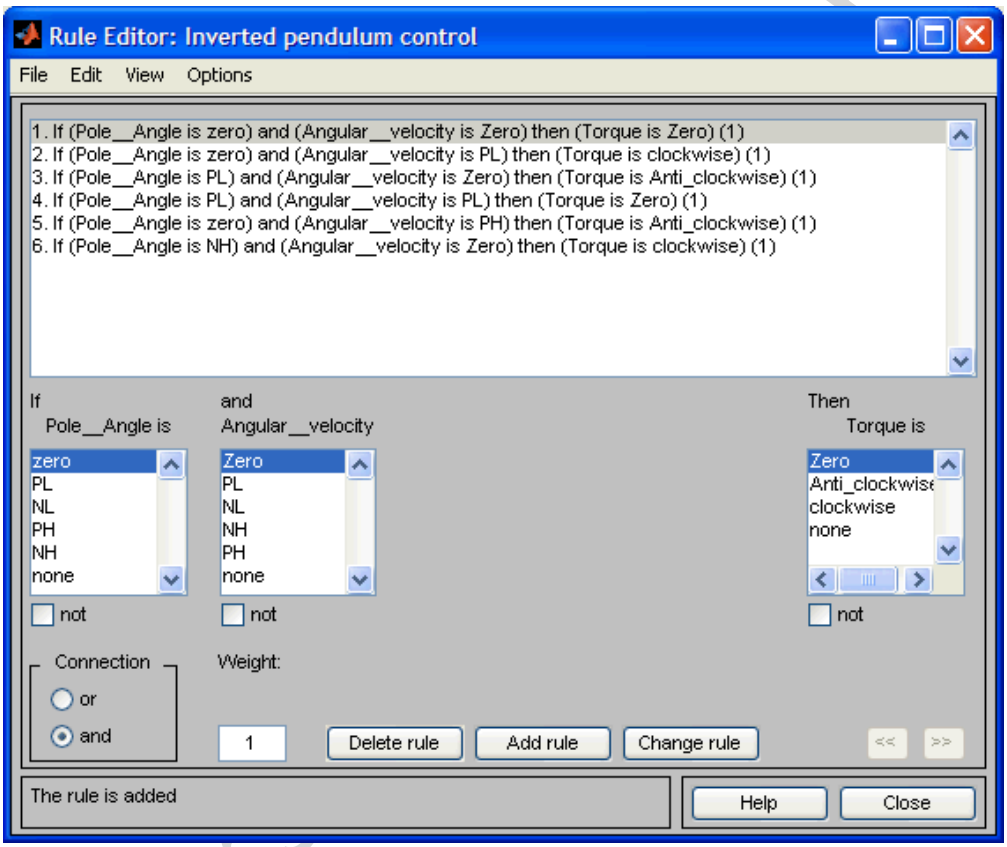


Figure 4.2.2 Show the fuzzy rules

4.2.3 Defuzzification

The defuzzifier takes the fuzzy interference system output and creates a single crispy output. There are several defuzzification methods, but probably the most popular one is the centroid technique. It finds the point where a vertical line would slice the aggregated set into two equal masses. Mathematically this centre of gravity (COG) can be expressed as [14]

$$COG = \frac{\int_a^b \mu_A(x)\, x\, dx}{\int_a^b \mu_A(x)\, dx}$$

Because of limited time i was unable to do FIS simulation, I have just created an example of fuzzy sets, membership functions and rules based on the input variables to the fuzzy system.

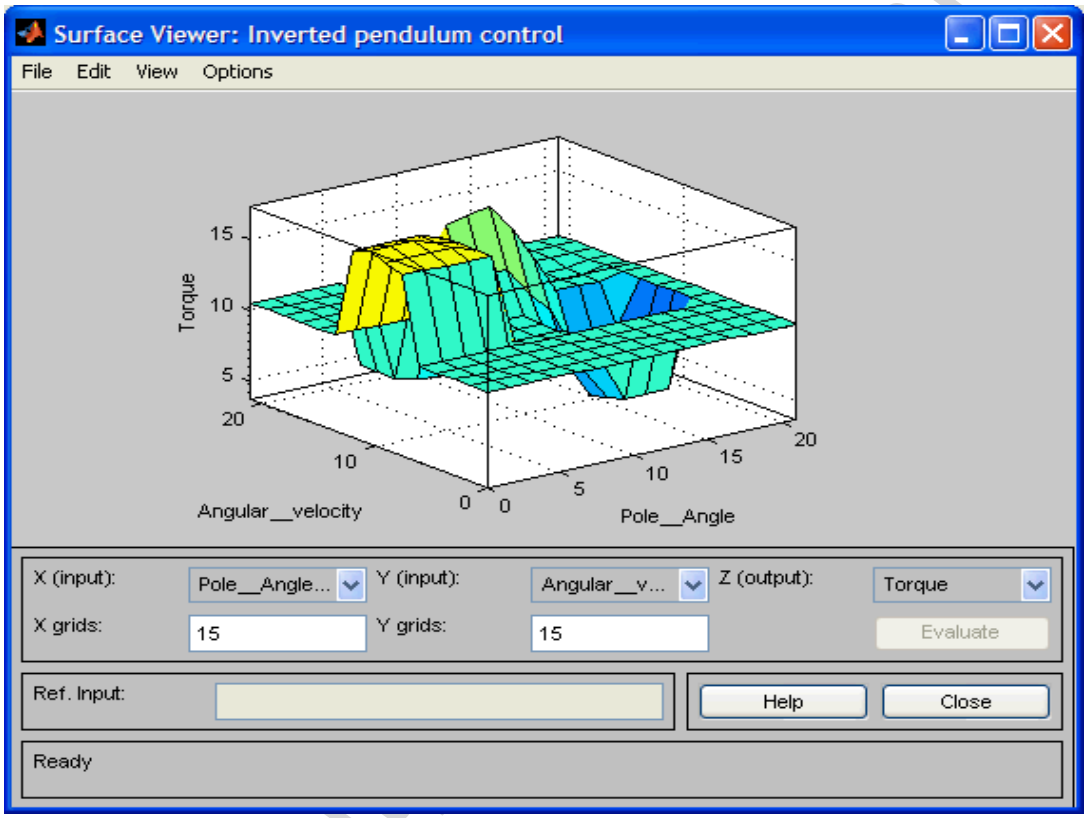


Figure 4.3.3 Surface of the fuzzy logic controller

University of Hertfordshire

5. Implementation and testing

In this chapter, inverted pendulum hardware is tested. The problems and difficulties faced for interfacing between sensor and Z8 controller are discussed.

5.1 Incremental encoder module

When I was trying to interface between Z8 encore controller and three channel incremental encoder module I faced a wiring problems. Because the first link (horizontal link) which is directly connected to the 2-phase bipolar stepper motor shaft is rotating by 360 degrees. I made the order for adjusting code wheel in incremental encoder but I haven't got it. So there is a slip between the code wheel and the shaft which is giving a bit noisy or distortion in the output signal. The output of the incremental encode is shown in figure 5.1.1

Figure 5.1.1 HEDS-9140 incremental output waveform from oscilloscope

5.2 Close the Inverted Pendulum Hardware

The signals generated by the three channel incremental encoder are send to the Z8 encoder development board port. Initially, I was planned to go for timers which will count the incoming pulses generated by the encoder and gives me the pole angle, pole position and velocity of the system.

Average velocity = (Position2 - Position1)/ time

Because of limited time period and technical problems, I was unable to do calculation of the inverted pendulum system. So finally I changed my mind to use interrupts. because if we look at the incremental encoder output waveforms, they are in quadrature (approximately 90

degree out of phase), mean if channel A leading channel B then the motor will be rotate in clockwise direction and if channel A lagging channel B then the motor will rotate in anti-clockwise direction. Therefore one interrupt is enough to know which channel is lagging and which channel is leading. So I wrote an interrupt service routine code which will simply look the 2-bit binary data on Z8 controller PortD based on that it will rotate the 2-ph bipolar stepper motor. Here I have chosen PortD pins (3 and 5) for Z8 controller input. I have chosen pin3 for interrupt.

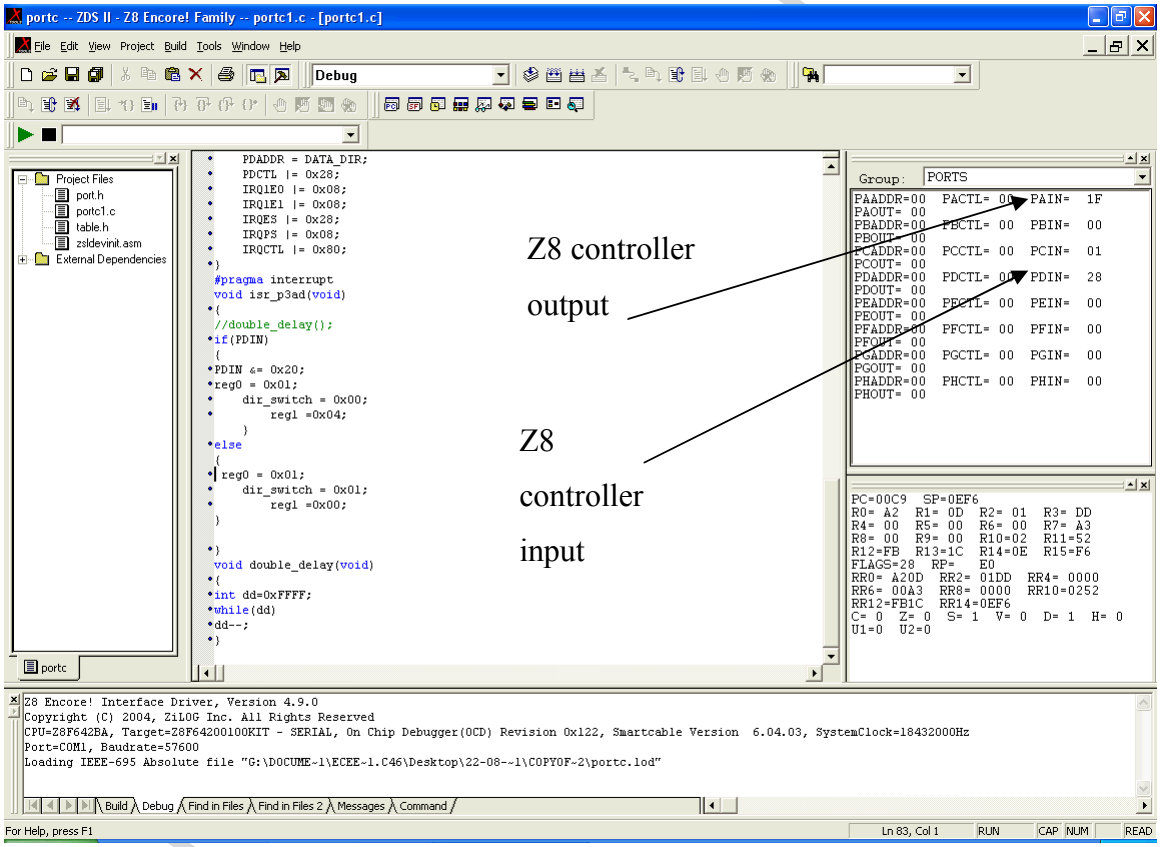


Figure 5.2.1 Software for controlling Inverted pendulum

Figure 5.2.1 and 5.2.2 shows how the values of the Z8 controller output Port A pin values are changing (PAOUT) with respect to the Z8 controller input PortD pins (PDIN).

```
if (PDOUT &= 0x20)
{
dir_switch() = 0x00
}
```

else

{

dir_switch () = 0x01

}

Every time an interrupt occurs the controller sees the PD5 (PortD pin 5) value if it low then the dir_switch value is 0x00 means the motor rotates in clockwise direction and if PD5 value is high then dir_switch is 0x01 hence the motor rotates in clockwise direction

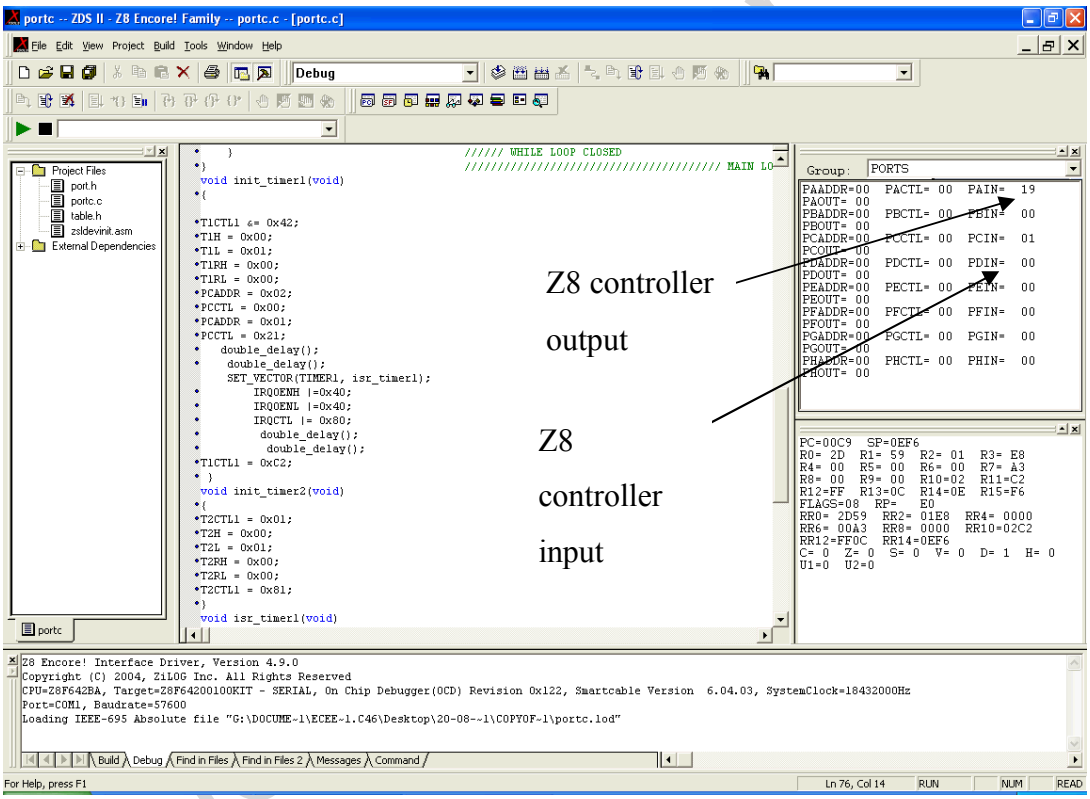


Figure 5.2.2 Software for controlling Inverted pendulum

The test was completed successfully and I have demonstrated the close loop inverted pendulum hardware to my supervisor.

7. Project Management

In this chapter, we will discuss the time management, budget management and resources in the lab for the inverted pendulum control project

7.1 Time Management

Inverted pendulum is a real-time control system. It need a lot of reading in different areas like mechanical, computer, electronics, electrical and control system in order to understand the fundamental and basic concepts. The actual time plan was quite close to the initial plan proposed, because of some technical and practical works there are some delays in time compare to the proposed time table

7.1.1 Extra time spend on selecting components and building inverted pendulum hardware

I struggled a lot for selecting inverted pendulum components because of low budget. When I was going for cheap components it is not compatible to the other components. A lot of research and test have made, I spend a lot of my project time on selecting the right components and building the inverted pendulum hardware. There is lots of delay in getting my components. Still I haven't got the code wheel alignment toolbox. Design and build the inverted pendulum hardware. Test the each part or component separately and then connect to the inverted pendulum hardware to avoid any damages. I have designed the supporting structure for inverted pendulum.

.

7.1.2 Extra time spend on programming, configuring the Z8 encore microcontroller development board.

In the initial time plan, the programming of Z8 controller is not included. This is one of the biggest challenging and time consuming area. A lot of time is spend on programming to interface between stepper motor, controller and PC. I have to look for C complier software provided by the ZDS II development board. Again because of technical problem as I said earlier I have to look for some other sort of resources. That is instead of incremental encoder I used PWM to generate a pulses which is similar to the incremental encoder module. Again I

University of Hertfordshire

have to look for its specifications and configurations so I used another microcontroller to generate pulses which has typically increased the hardware complexity.
I spend some time in learning C language; I learned different syntax of C language.
The software which is available in the electronic labs is not compatible to the Z8 controller development board so I have to install the appropriate software to interface between PC and Z8 controller.

7.2 Budget Management

This is another one of the challenging area for me. MSc students are allowed to spend 60£ budget for there projects. For example, If I am going for stepper motor and motor driver board it is only costing me around 55£ and more. However with the help of lab technician John Wilmot, I manage to get hardware for my project. Table 6.2.1 shows all the components used to build the inverted pendulum hardware.

Parts/components	Manufacturer	Supplier	Quantity x cost
Zilog Encore! Z8F64200100 development board	Zilog	UH Store	1 x £ 34.99
Three channel optical incremental encoder module	Hewlett Packard	Farnell	1 x £ 29.73
Three channel Code wheel	Hewlett Packard	Farnell	1 x £ 15.14
L298 Dual Full-Bridge Driver	National Semiconductor	Farnell	1 x £ 2.40
Bearings	RS	RS	2 x £ 3.78

University of Hertfordshire

HEDS-5140 code alignment tool	Hewlett Packard	Farnell	1 x £ 6.46
25 Pin IC socket for Z8 controller pins	-	UH store	1
Wooden Base	-	UH Store	1
2-phase bipolar stepper motor	Crouzet	UH Store	1
Supporting stand for Inverted pendulum Hardware with Pendulum	-	UH Store	1
6mm shaft (Fixed on first link)	-	UH Store	1
Total Cost			90.29

Table 6.2.1 Components and Cost

6.3 Equipments and Resources used in laboratory

- Two power supply is used, one for encoder (5V), Opto-coupler(5V) and L298 Dual Full-Bridge driver (12V)
- ZDS-II Z8 encore! controller software for debugging, compiling the C program
- An oscilloscope for waveform display
- Multimeter for testing circuit (voltage and current)
- Breadboards and wires for circuit connections

University of Hertfordshire

6. Conclusions and further work

This is the last chapter of the project report. In this chapter Project conclusion and further potential work for this project are highlighted.

6.1 Conclusion

The aim of the project is to design, build and implement FIS to the inverted pendulum hardware to balance a pole in upright position. The inverted pendulum hardware is successfully built. I have closed the inverted pendulum loop and I have tested this hardware. I observed how the direction of the inverted pendulum is changing based on the direction of the inverted pendulum to balance the pole in upright position. The basic idea of this control system is using a Z8 encore microcontroller which acts as an interface between inverted pendulum hardware and PC (software). The dynamically changing inverted pendulum variables (pole angle, position and direction) which are obtained from incremental encoder are send to the PC and simultaneously the output generated by the PC using C programming in ZDS II Z8 encore software is given to the Z8 encore controller, which will generate the 8-bit binary data for the stepper motor. Based on this 8-bit binary data stepper motor will rotate in either forward or backward direction, by giving the desired movement to the first link, the pendulum which is linked to this first link is kept on the upright position.

6.2 Achievements

Because of time limit, technical problems, limited resources, and practical problems, this project is partially completed. Even though I tried my best to complete and at the beginning everything was going smoothly but at the end of the project my hardware was not ready because of so many disturbances in the university.

The frame for the inverted pendulum is designed and build. This has taken plenty of time and one more issue is fix the incremental encoder on shaft which is directly connected to the pendulum. Practical work consumes a lot of time because everyday a new problem arises. For example when i was trying to interface between Z8 controller and H-bridge the output voltage on the Z8 controller is not sufficient for L298 dual full-bridge IC to drive 2-phase bipolar stepper motor. Hence Opto-coupler are used to increase the voltage. Finally I completed the inverted pendulum hardware, and I have tested this hardware. I have seen how the 8-bit of the Z8 controller are changing based on the inverted pendulum angle, which are obtained through

University of Hertfordshire

the incremental encoder and send to the Z8 microcontroller GPIO pins. Finally the inverted pendulum hardware is build and tested.

I have generate the two output waveforms (pulses) using two timers which are very similar to the incremental encoder output and I have calculated the pulse width modulation period in the mean time when my hardware was not ready, because the manufacturing department said they are not going to complete my hardware (supporting structure) because they have packed there stuff to move to some other place, because renovation or construction is taking place in the manufacturing department. John Wilmot helped me a lot, he requested in manufacturing department and finally i got the inverted pendulum hardware.

After I get the inverted pendulum as I said earlier I said one issue how to fix the incremental encoder on one end of the shaft, it has created a lot of trouble. We have to look what is the best possible way to fix this encoder on the shaft; finally we manage to fix the incremental encoder on the shaft. I ordered the code wheel alignment tool but till the end of the project I haven't got it.

Software part was the biggest challenge for me, because I don't know anything about the programming. I have gone through C programming books to learn about C coding. Dr David Lee helped me a lot in this software part and he explained me about the coding. Finally at the end I wrote a program to run a 2-phase bipolar stepper motor. Firstly I have tested this code on a breadboard and after seeing that the LEDs which are connected on the bread board are blinking in a correct sequence of pattern then I have giving this connections to the dual full bridge driver to run the motor. Finally I have tested the stepper motor; the 2-phase bipolar stepper motor changing its direction based on the 8-bit binary data presented to the stepper motor terminals for energizing the motor windings.

Furthermore, the study and research on Z8 controller to configure for this project is gone in much depth. This Z8 controller configuration is also one of the challenging part for this project. The suitable ports and pins have been chosen and assigned for particular purposes. The control and sub-registers for each port are studied to configure the Z8 encore controller for particular tasks like GPIO for receiving data and sending date from encoder to the stepper motor respectively, timers to generate output waveforms which are very similar to the incremental encoder output and interrupts to know which channel is leading and which channel is lagging. Finally, the connections of the incremental encoder, Z8 encore development board controller, motor and PC has been established.

6.3 Overall comment on this project

University of Hertfordshire

Even though FIS controller has not been developed and implemented on the real-time inverted pendulum hardware. By using a C code which is compiled and debugged in ZDS II C complier provided by the ZILOG Z8 encore development board, i control the pole of the inverted pendulum hardware. So finally we can say that for nonlinear systems like inverted pendulum can be controller by closing the loop of the system by using appropriate sensors like encoders. FIS controller are quite reliable for this type of systems because just by closing the loop of the inverted pendulum we cant get the desired position of the inverted pendulum in non-linear state.

6.4 Potential Further Work

6.4.1 Problems and Difficulties

Inverted pendulum hardware is completely built, but the only problem in this hardware is the friction between the first link and the shaft. As I said earlier manufacturing department was is in so hurry that they just build the hardware. Even though I used the bearing I was unable to reduce the friction between shaft and link. I think it will be fine if we change the shaft rod because the shaft rod is not smooth, I mean it has got some rough surface (bends).
Although the inverted pendulum hardware is tested using C programming compiled and debugged in ZDS II C compiler. But still I am not sure these coding work properly, therefore if required try to modify the program.

6.4.2 Further work

The performance of the inverted pendulum is limited by the low-budget. If a further work in this area is required, then following things have to changed
Instead of incremental encoder I will prefer the absolute encoders which give me the exact position of the shaft in one rotation. No need to count the pulses for calculating the pole position. Which will substantially decreases the burden on processor (Z8 controller) performance or hardware complexity (if we are using increment decoder/counter).
Although inverted pendulum hardware is completed, but still some modification have to be done to improve the performance of the system specially shaft which is fixed on the first link has to be changed to reduce the friction.
Design the FIS controller and implement this fuzzy interference system controller on the inverted pendulum hardware. Learn how to interface between host PC and Z8 controller using UARTs .

University of Hertfordshire

Reference:

1. Ye Zhang "Inverted pendulum with ANFIS controller", Project Report Master of Engineering, University of Hertfordshire, August 2005.
2. Kok Jiann Horng "Close the loop on the inverted pendulum hardware", Project Report Master of Engineering, University of Hertfordshire, September 2006.
3. T.Yamakawa, " Stabilization of an inverted pendulum by a high speed fuzzy logic controller hardware system", Fuzzy sets and systems,, Vol.32, no.2, pp 161-180,1989.
4. Data sheet, "2-Phase Bipolar Stepper Motor", http://www.crouzet.com/catalogue_web/int/eng/293/Stepper-motors-Direct-drive-stepper-motors-ENG-1201.htm
5. Data sheet " L298 Dual Full-bridge driver" http://www.st.com/stonline/products/literature/ds/1773/l298.pdfmcv
6. Data sheet "HEDS-9140 Incremental Encoder" http://www.farnell.com/datasheets/6186.pdf
7. Data sheet " HEDS-9140 Code Wheel" http://www.farnell.com/datasheets/16616.pdf
8. Dr. David Lee , " Neurofuzzy and cybernetics", University of Hertfordshire lecture notes, year 2007
9. Zilog application note: " A stepper motor controller using a ZILOG Encore! MCU", www.zilog.com
10. Perminder singh thiara, " DSP based fuzzy logic controller for an inverted pendulum", Final Year Report, University of Hertfordshire, April,2001.
11. Anderson, M.J and Grantham, W.J., " Lyapunov Optimal Feedback Control of a Nonlinear Inverted Pendulum",Journal of Dynamic Systems, Measurement and Control, Vol.111,pp.554-558, 1989.
12. Sheng Liu,Lihui Cui,Jie Chen and Ming Bai, " Research of Rotary Inverted Pendulum Using Fuzzy Strategy Based on Dynamic", Proceedings of the 4th world congress on intelligent control and automation, Shanghai, P.R, China june 2002.
13. Micheal Negnevitsky, " Artificial Intelligence- A Guide to Intelligent systems", first edition,2002.
14. Lily Meng , " Neurofuzzy and cybernetics" ,University of Hertfordshire lecture notes, year 2007
15. Zilog product specification, "Z8 encore! 64K series High performance 8-bit microcontroller product specifications". www.zilog.com

University of Hertfordshire

Bibligraphy:

1. Chen Wei Ji, Fang Lei and Lei Kam Kin, " Fuzzy logic controller for an inverted pendulum system", proceedings of the IEEE International conference on Intelligent Processing Systems, Beijing, China, pp.185-189, October 1997.
2. Mr. James Driver and Mr. Dylan Thorpe "Design build and control of a single/double rotational inverted pendulum" Final year report of engineering, The university of Adelaide school of Mechanical Engineering, October 2004.
3. Sigeru Omatu, Michifumi Yoshioka, "Stability of Inverted pendulum by Neuro-PID control with Genetic Algorithm",IEEE: 0-7803-4859-1/98,pp: 2142-2145,1998
4. Zilog Technical note, "Using the GPIO pins of the Z8 Encore! MCU" TN002401-0304, www.zilog.com
5. BalaguruSamy "Programming in Ansi C", McGraw-Hill Education, march 2004.
6. The Math Works, Inc. www.mathworks.com
7. C Programming www.cprogramming.com

University of Hertfordshire

Appendix A: C Program for Z8 Controller

```
#include<ez8.h>
#include"port.h"

////////////////////// definitions of Port///////////////

#define DATA_DIR 0x01;
#define ALT_FUN  0x02;
#define OUT_CTL  0x03;

/////////////// Global Declarations//////////
int dir_switch;
int reg0,reg1;
void isr_p3ad(void);
void init_p3ad(void);

///////////////// main routine//////////
void main()
{
        PAADDR=ALT_FUN;
        PACTL = 0x00;
        PAADDR= DATA_DIR;
        PACTL = 0x00;
        PAADDR = OUT_CTL;
        PACTL = 0x00;
        PAOUT = 0x00;
        init_p3ad();

while(1)
        {
        double_delay();
if (reg0 == 0x00)
```

University of Hertfordshire

```
        {
        PAOUT = 0x00;
        }
else
        {
        reg0 == 0x01;
        if(dir_switch)
        {
                reg1++;
                if(reg1>4)
                {
                        reg1=1;
                }
        }
        else
        {
                reg1--;
                if(reg1<1)
                {
                        reg1=4;
                }

        }
        switch(reg1)          ////////// Pattern of energizing the motor windings
        {
                case 1:
                        PAOUT = 0x05;
                        break;
                case 2:
                        PAOUT = 0x06;
                        break;
                 case 3:
                        PAOUT = 0x0A;
```

University of Hertfordshire

```
                    break;
                case 4:
                    PAOUT  =0x09;
                    reg1 = 0;
                    break;
                    }                    ////////////// SWITCH LOOP CLOSED ////////////////
}                   /////ELSE LOOP CLOSED//////////////////////////
        }                   ////// WHILE LOOP CLOSED//////////////////////////
}                   ////////////////////////////////////// MAIN LOOP CLOSED//////////////////



///////////////// initializes the P3AD////////////////
void init_p3ad(void)
{
        SET_VECTOR(P3AD, isr_p3ad);
{
        SET_VECTOR(P3AD, isr_p3ad);
        PDADDR = ALT_FUN;
        PDCTL = 0x00;
        PDADDR = DATA_DIR;
        PDCTL |= 0x28;
        IRQ1E0 |= 0x08;
        IRQ1E1 |= 0x08;
        IRQES |= 0x08;
        IRQPS |= 0x28;
        IRQCTL |= 0x80;
}

/////////////// interrupt service routine for P3AD////////
#pragma interrupt
void isr_p3ad(void)
{
```

University of Hertfordshire

```
reg0 = 0x01;
if(PDIN)
{
        PDIN &= 0x20;
        dir_switch = 0x00;      /////// Motor will rotate in anti-clockwise direction//////////
                }
else
{
        dir_switch = 0x01;      ///////// Motor will rotate in clockwise direction/////////
}

}
/////////////// provides delay //////////////
void double_delay(void)
{
int dd=0xFFFF;
while(dd)
dd--;
}
```

Code for generating Pulses using Timer1 and Timer2

```
#include <eZ8.h>
#include<stdio.h>
void init_timer0(void);
void init_timer1(void);
void main()
{   PAADDR = 0x02;
        PACTL = 0xFF;
        PCADDR = 0x02;
        PCCTL = 0xFF;
        init_timer0();
```

University of Hertfordshire

```
        init_timer1();
        T0CTL |= 0x80;
        while(1);
}
        void init_timer0(void)
 {
        T0CTL1 = 0xFB;
        T0H = 0x00;
        T0L = 0x01;
        T0PWMH = 0x80;
        T0PWML = 0x00;
        T0RH = 0x00;
        T0RL = 0x00;
        deb_delay();
        deb_delay();
//      T0CTL1 = 0x3C;
  }
 void init_timer1(void)
 {

        T1CTL1 = 0xFB;
        T1H = 0x00;
        T1L = 0x01;
        T1PWMH = 0x80;
        T1PWML = 0x00;
        T1RH = 0x00;
        T1RL = 0x00;
//      T1CTL1 = 0x3C;
        deb_delay();
        deb_delay();
 }
 void deb_delay()
        {
```

University of Hertfordshire

```
int dd = 0xFFFF;
while (dd)
        dd--;
        }
```

University of Hertfordshire

Appendix B: Motor specifications

Direct drive stepper motors

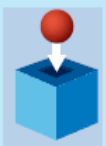
➜ 7.5° 10 Watts

- 48 steps/revolution (7.5°)
- Absorbed power : 10 W
- 2 or 4 phase versions available

Specifications

			2 phases	4 phases
Type			82 930 0	82 930 0
Number of phases			2	4
Electronic controller used			Bipolar	Unipolar
Resistance per phase (Ω)	**Current per phase (A)**	**Voltage at motor terminals (V)**		
9	0,75	6,6	●	
22,3	0,48	10,4	**82 930 002**	
7,4	0,81	6		●
32	0,39	12,5		**82 930 015**
General characteristics				
Absorbed power (W)			10	10
Holding torque (mN.m)			180	155
Step angle (°)			7.5	7.5
Positioning accuracy (mm)			5	5
Inertia of rotor (gcm²)			84	84
Max. detent torque (mN.m)			12	12
Max. coil temperature (°C)			120	120
Storage temperature (°C)			-40 → +80	-40 → +80
Thermal resistance of coil - ambient air (°C/W)			7	7
Insulation resistance (at 500 Vcc) (MΩ) following NFC 51200 standard			> 10^3	> 10^3
Insulation voltage (50 Hz, 1 minute) (V) following NFC 51200 standard			> 600	> 600
Wires length (mm)			250	250
Weight (g)			340	340
Protection rating			IP40	IP 40

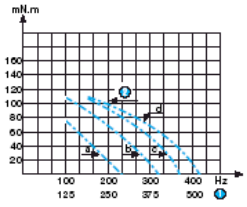
Product adaptations



- Special output shafts
- Special supply voltages
- Special cable lengths
- Special connectors

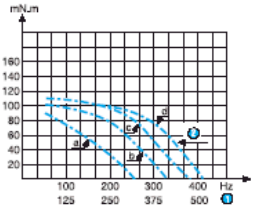
Curves

2 phases



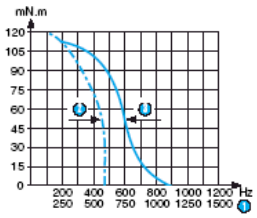
1 RPM
2 Max. stopping-starting curves

4 phases



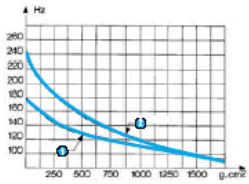
1 RPM
2 Max. stopping-starting curves

2 phases - Max. stopping-starting and operating curves at I constant (PBL 3717) for 2 (motor) phases 9 Ω. Holding torque 150 mN.m. Current per phase 0.53 A



1 RPM
2 Max. stopping-starting curves
3 Max. operating curves

Max. stopping-starting frequency curves as a function of the external inertia load at zero antagonistic torque. Tests at constant U.

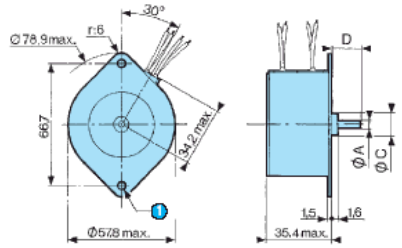


1 2 phases
2 4 phases

Inertia of measuring chain : 3.4 g.cm²
a = constant voltage controller with Rs (resistance in series) = 0
b = constant voltage controller with Rs (resistance in series) = R motor
c = constant voltage controller with Rs (resistance in series) = 2R motor
d = constant voltage controller with Rs (resistance in series) = 3R motor

The measurements are made with full stepping, 2-phases energised.

Dimensions



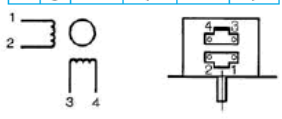
Axe version	Ø A	Ø C	D
Version 1	$4^{\ 0}_{-0.008}$	$12^{\ 0}_{-0.05}$	16
Version 2	$6{,}35^{\ 0}_{-0.01}$	$11{,}13^{\ 0}_{-0.05}$	16
Version 3	$6{,}35^{\ 0}_{-0.01}$	$12{,}7^{\ 0}_{-0.05}$	16

1 2 Fixing holes Ø 4.4

Connections

2 phases

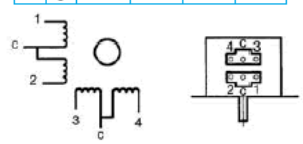
		1	2	3	4
1	1	-	+	-	+
	2	-	+	+	-
	3	+	-	+	-
	4	+	-	-	+
	5	-	+	-	+



1 Step
Energisation sequence for clockwise rotation : (viewed shaft end)

4 phases

		1	2	3	4
1	1	-		-	
	2	-			-
	3		-		-
	4		-	-	
	5	-		-	



1 Step
Energisation sequence for clockwise rotation : 2 phases energised (viewed shaft end, front forward)
Commons connected to positive.

Appendix C: HEDS-9140 incremental encoder and code wheel specifications

Three Channel Optical Incremental Encoder Modules

Technical Data

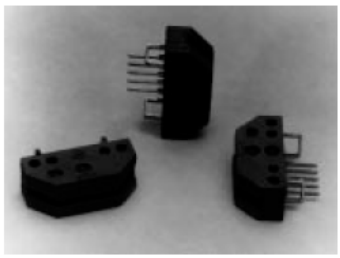
HEDS-9040
HEDS-9140

Features

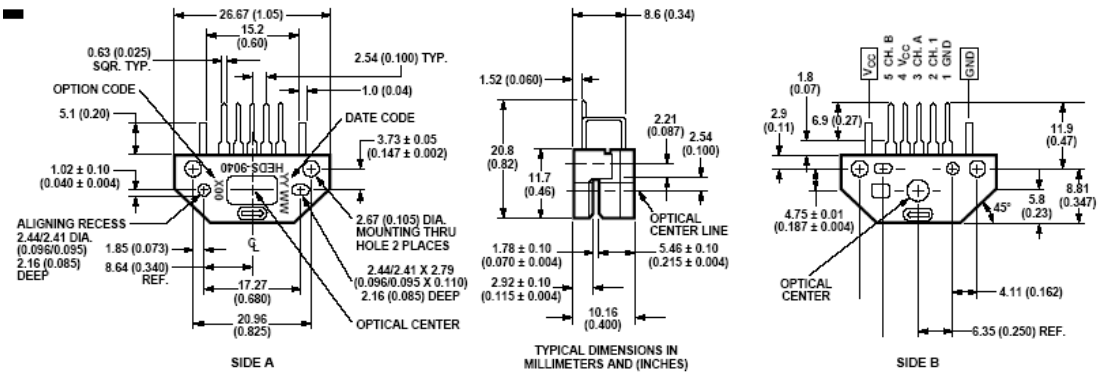
- Two Channel Quadrature Output with Index Pulse
- Resolution Up to 2000 CPR Counts Per Revolution
- Low Cost
- Easy to Mount
- No Signal Adjustment Required
- Small Size
- -40°C to 100°C Operating Temperature
- TTL Compatible
- Single 5 V Supply

Description

The HEDS-9040 and HEDS-9140 series are three channel optical incremental encoder modules. When used with a codewheel, these low cost modules detect rotary position. Each module consists of a lensed LED source and a detector IC enclosed in a small plastic package. Due to a highly collimated light source and a unique photodetector array, these modules provide the same high performance found in the HEDS-9000/9100 two channel encoder family.



Package Dimensions



Recommended Operating Conditions

Parameter	Symbol	Min.	Typ.	Max.	Units	Notes
Temperature	T_A	-40		100	°C	
Supply Voltage	V_{CC}	4.5	5.0	5.5	Volts	Ripple < 100 mV_{p-p}
Load Capacitance	C_L			100	pF	2.7 kΩ pull-up
Count Frequency	f			100	kHz	Velocity (rpm) x N/60
Shaft Perpendicularity Plus Axial Play				± 0.25 (± 0.010)	mm (in.)	6.9 mm (0.27 in.) from mounting surface
Shaft Eccentricity Plus Radial Play				0.04 (0.0015)	mm (in.) TIR	6.9 mm (0.27 in.) from mounting surface

Note: The module performance is guaranteed to 100 kHz but can operate at higher frequencies. For the HEDS-9040 #T00 for operation below 0°C and greater than 50 kHz the maximum Pulse Width and Logic State Width errors are 60°e.

d

University of Hertfordshire

Mounting with an Alignment Tool

The HEDS-8905 and HEDS-8906 alignment tools are recommended for mounting the modules with Side B as the mounting plane. The HEDS-8905 is used to mount the HEDS-9140, and the HEDS-8906 is used to mount the HEDS-9040. These tools fix the module position using the codewheel hub as a reference. They will not work if Side A is used as the mounting plane.

The following assembly procedure uses the HEDS-8905/8906 alignment tool to mount a HEDS-9140/9040 module and a HEDS-5140/6140 codewheel:

Instructions:

1. Place codewheel on shaft.

2. Set codewheel height by placing alignment tool on motor base (pins facing up) flush up against the codewheel as shown in Figure 3. Tighten codewheel setscrew and remove alignment tool.

3. Insert mounting screws through module and thread into the motor base. Do not tighten screws.

4. Slide alignment tool over codewheel hub and onto module as shown in Figure 4. The pins of the alignment tool should fit snugly inside the alignment recesses of the module.

5. While holding alignment tool in place, tighten screws down to secure module.

6. Remove alignment tool.

Mounting with Aligning Pins

The HEDS-9040 and HEDS-9140 can also be mounted using aligning pins on the motor base. (Hewlett-Packard does not provide aligning pins.) For this configuration, Side A must be used as the mounting plane. The aligning recess centers must be located within the 0.005 in. R Tolerance Circle as explained above. Figure 5 shows the necessary dimensions.

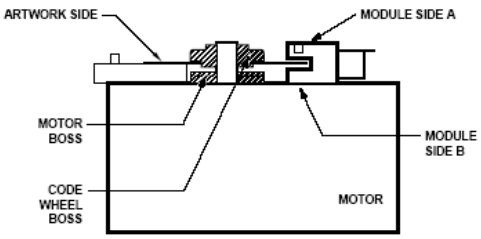


Figure 3. Alignment Tool is Used to Set Height of Codewheel.

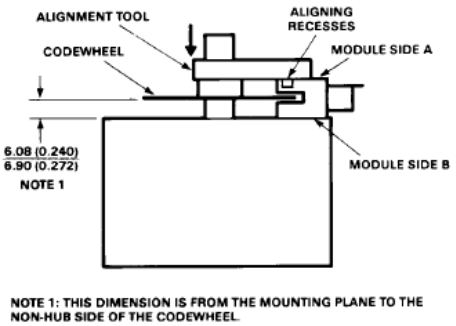
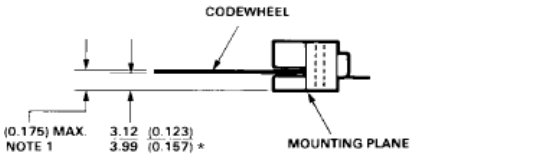


Figure 4. Alignment Tool is Placed over Shaft and onto Codewheel Hub. Alignment Tool Pins Mate with Aligning Recesses on Module.

Univ

Mounting with Aligning Pins

The HEDS-9040 and HEDS-9140 can also be mounted using aligning pins on the motor base. (Agilent does not provide aligning pins.) For this configuration, Side A *must* be used as the mounting plane. The aligning recess centers must be located within the 0.005 in. Radius Tolerance Circle as explained in "Mounting Considerations." Figure 5 shows the necessary dimensions.



NOTE 1: THESE DIMENSIONS INCLUDE SHAFT END PLAY AND CODEWHEEL WARP.
NOTE 2: RECOMMENDED MOUNTING SCREW TORQUE IS 4 KG-CM [3.5 IN-LBS].
*FOR HEDS-9040 OPTION T: 3.99 (0.150).

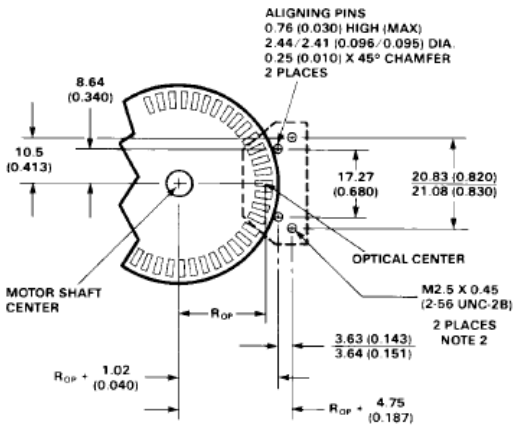


Figure 5. Mounting Plane Side A.

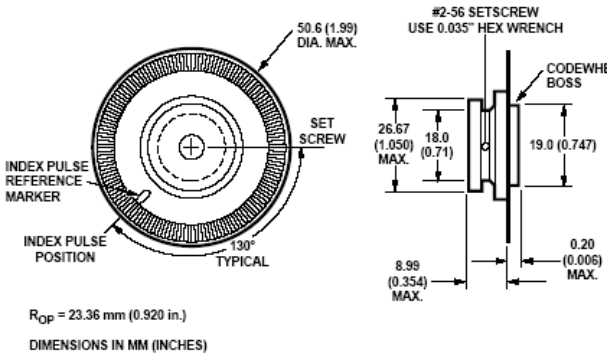


Figure 6a. HEDS-6140 Codewheel Used with HEDS-9040.

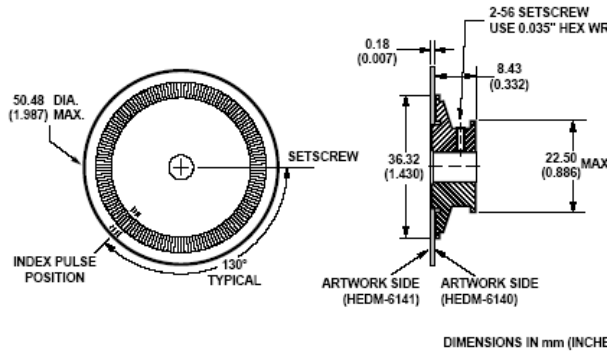


Figure 6b. HEDM-614X Series Codewheel used with HEDS-9040

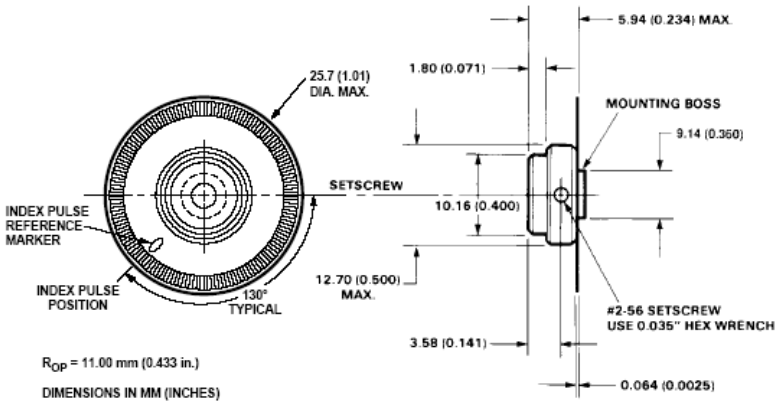
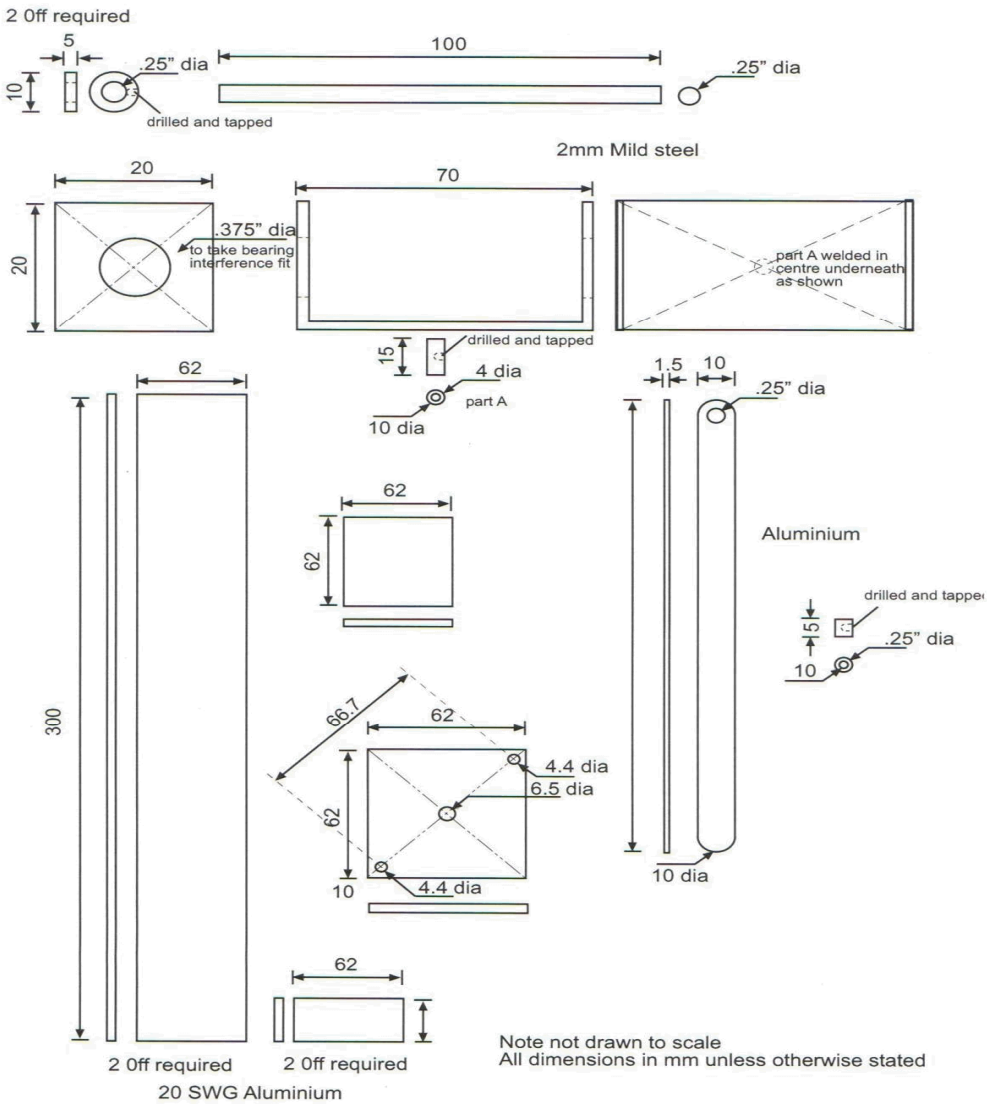
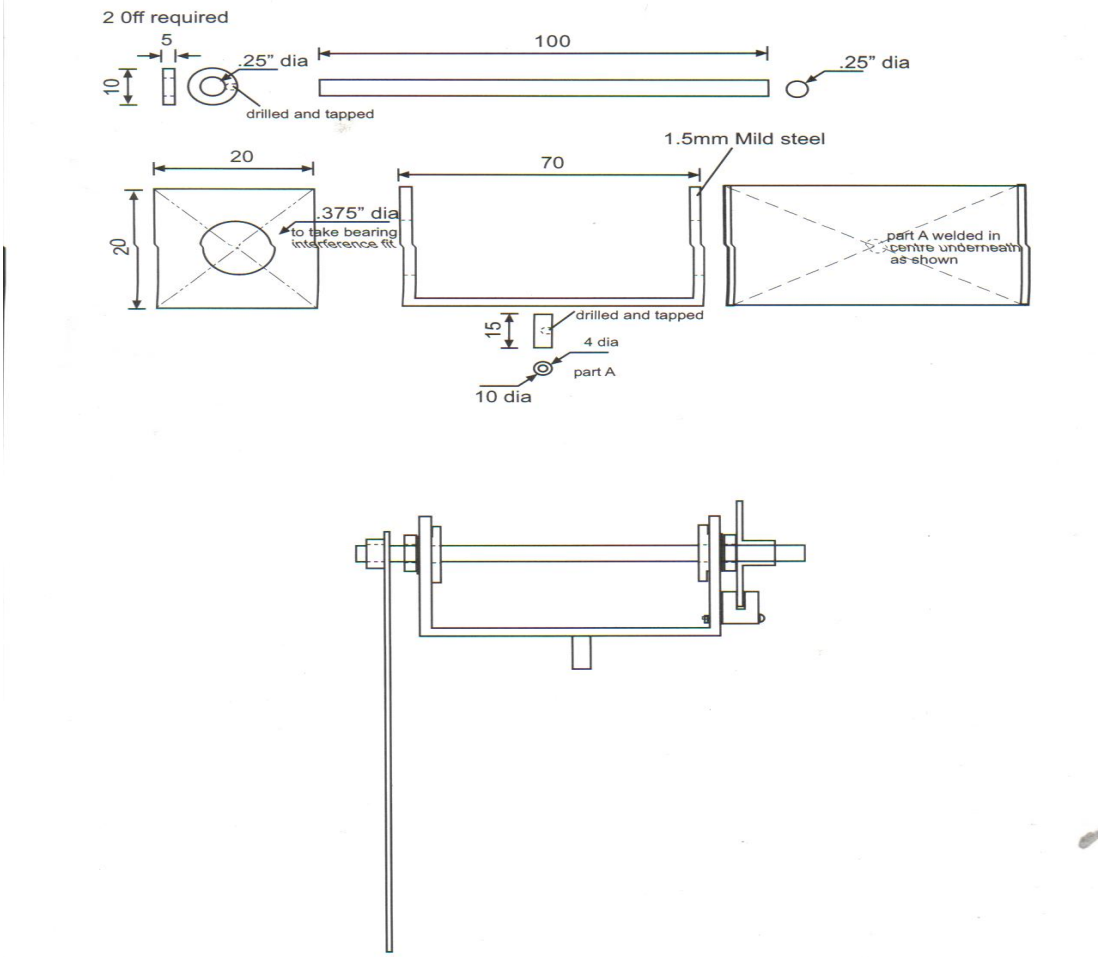


Figure 7. HEDS-5140 Codewheel Used with HEDS-9140.

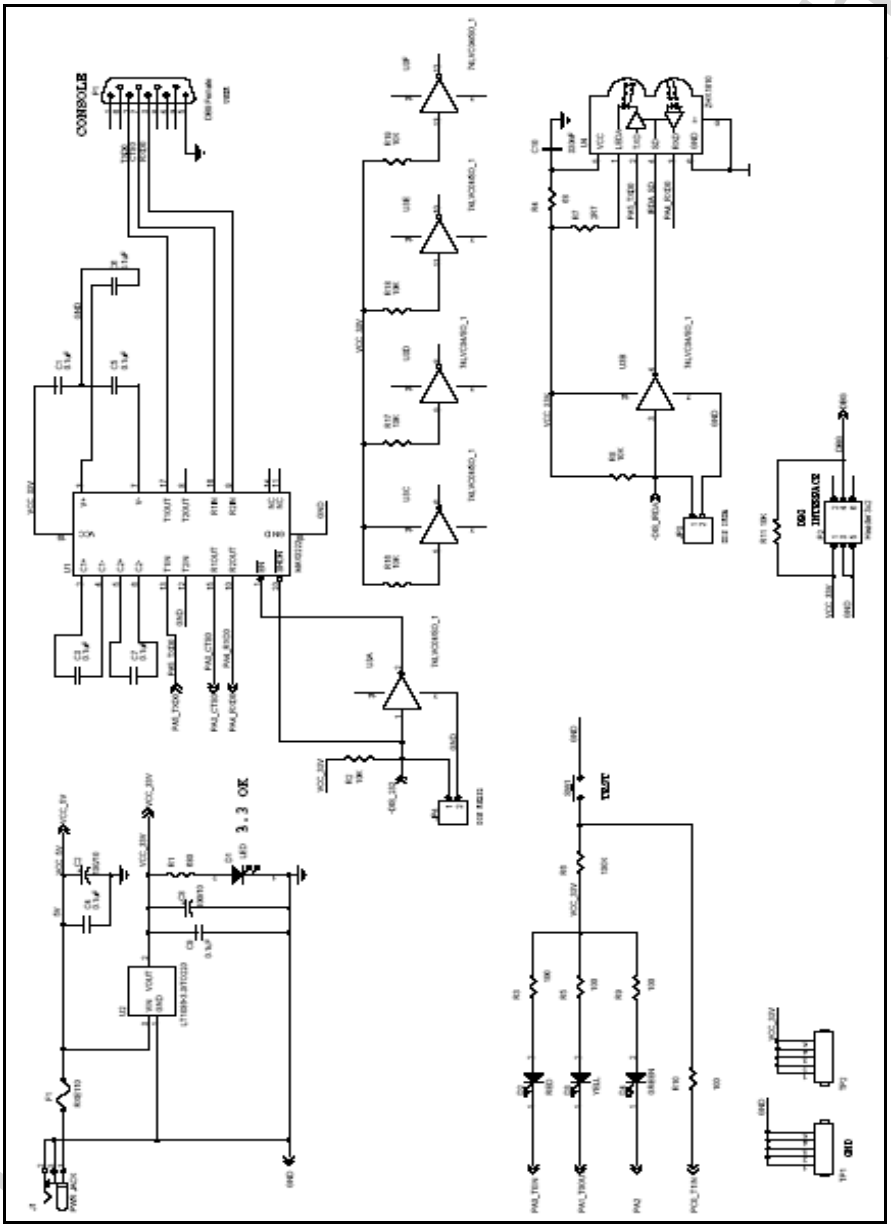
Appendix D: Mechanical Drawings

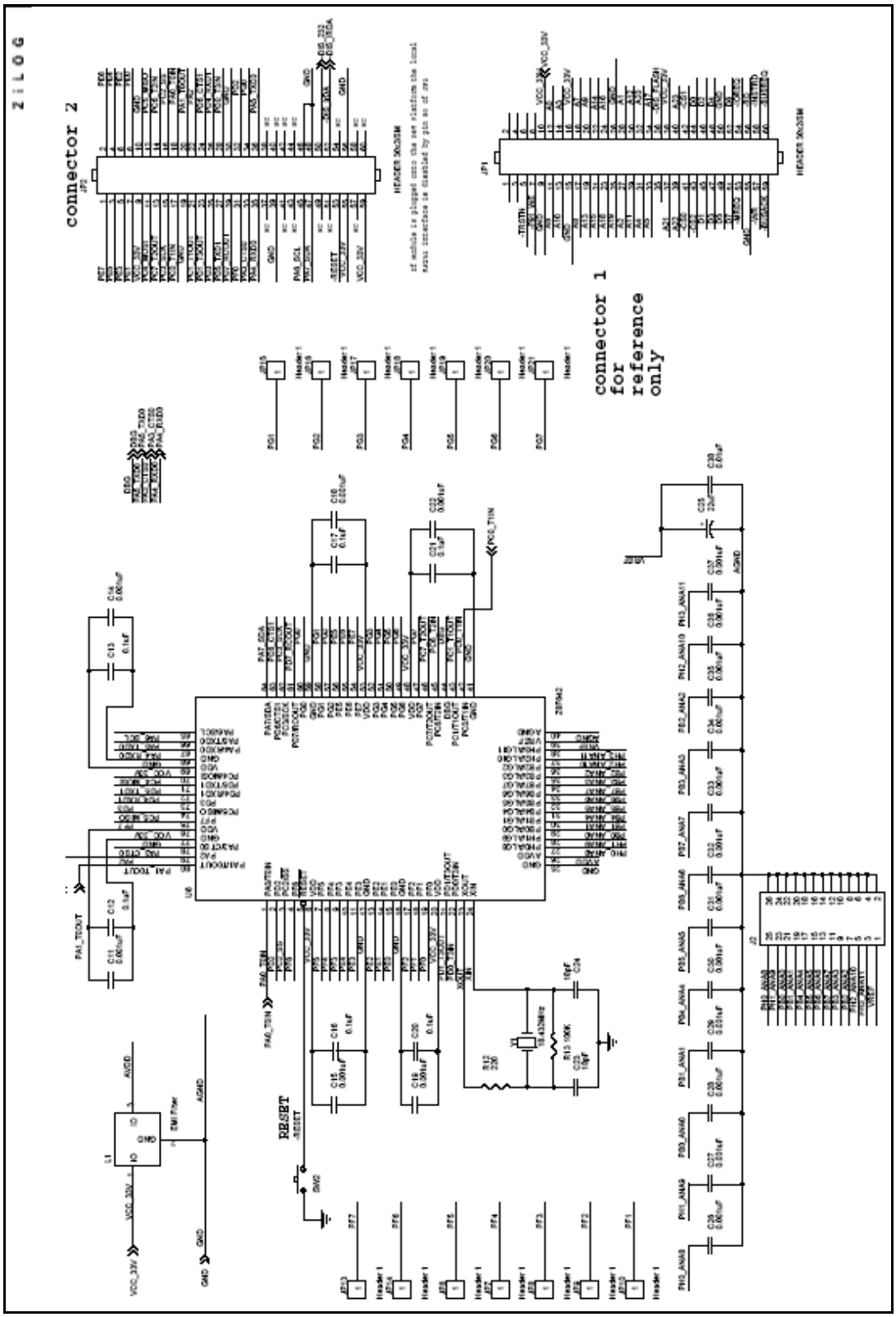


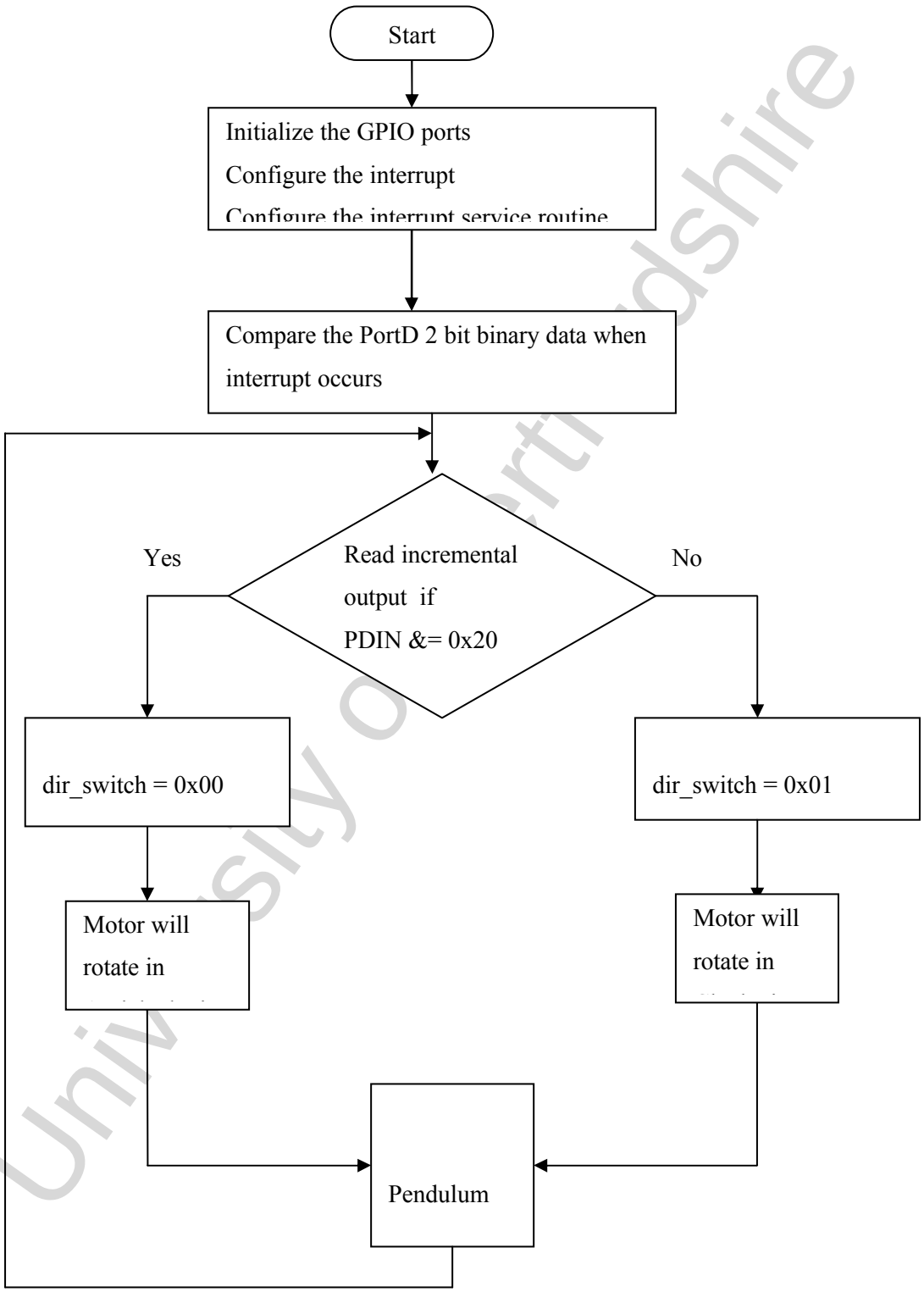

2 0ff required
5
.25" dia
100
.25" dia
10
drilled and tapped
20
1.5mm Mild steel
70
.375" dia
to take bearing
interference fit
20
part A welded in
centre underneath
as shown
drilled and tapped
15
4 dia
part A
10 dia

Univer

Appendix E: Schematics for the Z8 encore! 64K MCU Development board




ZiLOG
connector 2
connector 1
for
reference
only
RESET



University of Hertfordshire